Universitat Oberta de Catalunya
uoc.edu

“Implementación de sistema SSO basado en Shibboleth”

Alumno: Juan Fco Rodríguez Moreno
Plan de estudios: Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones. MISTIC
Sistemas de autenticación y autorización

Director del TFM: Antoni Gonzalez Ciria
Profesor/a responsable de la asignatura: Victor Garcia Font

Fecha de entrega: 31 de diciembre de 2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	Implementación de sistema SSO basado en Shibboleth
Nombre del autor:	Juan Fco Rodríguez Moreno
Nombre del consultor/a:	Antoni Gonzalez Ciria
Nombre del PRA:	Victor Garcia Font
Fecha de entrega (mm/aaaa):	12/2019
Titulación::	Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones
Área del Trabajo Final:	Sistemas de autenticación y autorización
Idioma del trabajo:	Español
Palabras clave	SSO, Sistemas de autenticación única, Shibboleth

Resumen del Trabajo

En un escenario con una gran variedad de servicios online, los sistemas de autenticación única (SSO) permiten simplificar la gestión de identidades. El objetivo de este trabajo es la construcción de un sistema de autenticación única basado en estándares abiertos y en productos de código abierto.

El sistema implementado permite verificar la identidad de los usuarios mediante mecanismos basados en usuario-contraseña y certificado electrónico. Además, el sistema recupera información del usuario, desde el repositorio de la organización, y la pone a disposición de las distintas aplicaciones. Se tiene especial cuidado en el control, por parte del usuario, de la cesión de sus datos personales. Por último, se han considerado aspectos de seguridad en las distintas fases del desarrollo del sistema.

El sistema implementado cumple los requerimientos establecidos, si bien hay distintos aspectos de mejora. Estos aspectos están relacionados principalmente con la disponibilidad de distintos componentes del sistema, como el proveedor de identidad o el servicio de directorio, y otros relacionados con los mecanismos de autorización y personalización del sistema.

Abstract

In the current scene with wide variety of online services, single sign on systems (SSO) allow simplify the identities management. The aim of this work is to develop a single sign on system based on open standards and open source products.

The deployed system allows verifying user identity through mechanisms based on user-password and digital certificate. In addition, the system retrieves user information from organization repository and makes it available to applications. Special care is applied to the control, by the user, the transfer of their personal data. Finally, security aspects have been considered throughout all the stages of the project.

The system meets the established requirements, although various improvements are proposed. These are mainly related to the availability of the components of the system, such as the identity provider or the directory service, and other aspects such as system authorization and customization.

Índice de contenidos

Índice de contenidos 2
Índice de figuras 3
Índice de tablas 4
1. Introducción 5
1.1 Problema a resolver 5
1.2 Objetivos 5
1.3 Metodología 6
1.4 Planificación 7
1.3 Organización de la memoria 9
2. Estado del arte 10
2.1 Autenticación/autorización 10
2.2 Administración de identidades 10
2.3 SSO 11
2.4 SAML v2 13
2.5 Shibboleth 18
2.6 Alta disponibilidad 23
3. Solución propuesta 27
3.1 Análisis de requisitos 27
3.2 Análisis 29
3.3 Arquitectura del sistema 35
3.4 Diseño detallado e implementación 36
3.4 Validación 52
4. Conclusiones 55
4.1 Conclusiones del Trabajo 55
4.2 Mejoras / Futuras aportaciones 56
Bibliografía 59
Anexos 60
Anexo A. Requerimientos de Shibboleth 60
Anexo B. Infraestructura de red desplegada 61

Índice de figuras

Figura 1 - Fases del modelo iterativo y creciente Fuente:https://en.wikipedia.org/wiki/Ite 7
Figura 2 - Lista de tareas definidas en la planificación del proyecto 8
Figura 3 - Planificación temporal del proyecto 8
Figura 4 - Conceptos básicos de SAML. Fuente: SAML V2.0 Technical Overview 14
Figura 5 - Flujo en perfil Web SSO iniciada por SP. Fuente: SAML V2.0 Technical Overview 17
Figura 6 - Arquitectura de proveedor de identidad de Shibboleth. Fuente: https://wiki.shib ... 19
Figura 7 - Diagrama de Caso de Uso CU01 29
Figura 8 - Arquitectura del sistema 35
Figura 9 - Diagrama de red del sistema 35
Figura 10 - Modelo lógico de datos del servicio de directorio 36
Figura 11 - Cliente Ldap (LDAP Admin) 37
Figura 12 - Diagrama de componentes de Shibboleth IdP3 38
Figura 13 - Diagrama de componentes del proveedor de servicios 41
Figura 14 - Diagrama del interfaz de usuario de las aplicaciones web 43
Figura 15 - Diagrama de componentes del servidor de aplicaciones 43
Figura 16 - Página de inicio (zona pública) de la aplicación proveedores 44
Figura 17 - Página de inicio de la aplicación experiencias 44
Figura 18 - Página de inicio de la aplicación proveedores 44
Figura 19 - Certificado de usuario generado para joyceb 46
Figura 20 - Logout del proveedor de servicios 48
Figura 21 - Logout del proveedor de identidad 48
Figura 22 - Diagrama de red del sistema 51
Figura 23 - Página del área pública de la aplicación de Gestión de proveedores 52
Figura 24 - Pagina de login del IdP 52
Figura 25 - Página de inicio de la aplicación proveedores 52
Figura 26 - Mensaje de error en página de login tras error de autenticación 53
Figura 27 - Mensaje de error en página de login tras error en la contraseña 53
Figura 28 - Página de login con certificado electrónico y selección del mismo 53
Figura 29 - Página de logout de la aplicación 53
Figura 30 - Pagina de login (forzando pedir autorización) 54
Figura 31 - Pagina de concesión de autorización 54

Índice de tablas

Tabla 1 - Descripción del escenario e01 del caso de uso CU01 29
Tabla 2 - Descripción del escenario e02 del caso de uso CU01 30
Tabla 3 - Descripción del escenario e03 del caso de uso CU01 30
Tabla 4 - Descripción del escenario e04 del caso de uso CU01 31
Tabla 5 - Especificación del caso de prueba CP01..... 31
Tabla 6 - Especificación del caso de prueba CP02..... 31
Tabla 7 - Especificación del caso de prueba CP03..... 32
Tabla 8 - Especificación del caso de prueba CP04..... 32
Tabla 9 - Especificación del caso de prueba CP05..... 32
Tabla 10 - Especificación del caso de prueba CP05..... 32
Tabla 11 - Especificación del caso de prueba CP06..... 33
Tabla 12 - Especificación del caso de prueba CP07..... 33
Tabla 13 - Usuarios definidos en el sistema..... 34
Tabla 14 - Relación de usuarios / roles del sistema 34
Tabla 15 - Lista de máquinas virtuales desplegadas 51
Tabla 16 - Lista de subredes del sistema..... 61
Tabla 17 - Lista de subredes de gestión 61

Y los galaaditas tomaron los vados del río Jordán a Efraím, y cuando alguno de los de Efraím que había huido decía: «¿Pasaré?». Los de Galaad le preguntaban: «¿Eres tú efrateo?». Si él respondía «no», entonces le decían: «Pues di "shibboleth"».
Y él decía «sibboleth», porque no podía pronunciar aquella suerte.
Entonces le echaban mano y le degollaban. Y así murieron cuarenta y dos mil de los de Efraím."
Libro de los jueces, capítulo 12, versículos 4 a 6

1. Introducción

1.1 Problema a resolver

En nuestro día a día utilizamos un número cada vez mayor de servicios online como redes sociales, correo electrónico u aplicaciones empresariales entre otros, en los que la verificación de la identidad de los usuarios requiere el uso de contraseñas. Esto conlleva que sea frecuente tener múltiples credenciales con las acceder a los distintos servicios. Al aumentar el número de servicios, la gestión de las credenciales se vuelve cada vez más complicada. Esta situación puede generar fatiga de contraseña o de autenticación, al tener que memorizar un excesivo número de contraseñas. Esta situación puede dar lugar a seguir conductas que reducen la seguridad de los sistemas de información, como utilizar escoger contraseñas fáciles de recordar o guardar por escrito las contraseñas.

Los sistemas de administración de identidades tratan de facilitar la gestión de identidades de un usuario. Uno de sus componentes son los mecanismos de autenticación única o Single Sign On (SSO). Estos permiten acceder a distintos servicios o aplicaciones con las mismas credenciales. Una vez que un usuario se ha autenticado para acceder a un servicio, y durante un tiempo limitado, puede acceder a otros servicios o aplicaciones sin tener que autenticarse de nuevo. De esta forma se consigue reducir la fatiga de autenticación al reducir el número de credenciales a recordar, se aumenta la usabilidad del sistema y se disminuye la probabilidad de seguir conductas que disminuyan la seguridad del mismo.

1.2 Objetivos

El objetivo de este trabajo es la construcción de un sistema de autenticación única (SSO) basado en estándares abiertos utilizando productos de código abierto. Además el sistema ha de cumplir ciertos requisitos iniciales.

Respecto a los requisitos iniciales, la solución debe permitir controlar el acceso de los usuarios a los recursos o aplicaciones a través de dos mecanismos de autenticación, basados en usuario-contraseña y certificado electrónico.

Se desplegaran un conjunto de aplicaciones con las que se podrá verificar que el sistema implementado cumple con los requerimientos indicados.

Respecto a las aplicaciones, debe ser posible recuperar información relativa al usuario almacenada en el repositorio de la organización, ya sea en un servicio de directorio, en una base de datos o de otro tipo. El usuario ha de ser consciente y autorizar la cesión de sus datos personales que se realice a las aplicaciones o al proveedor de servicios.

Por último, se deben establecer las medidas de seguridad necesarias, tanto en la infraestructura de red como en los distintos componentes del sistema.

1.3 Metodología

Se han analizado distintas metodologías software para elegir la que resulte más adecuada para la construcción del sistema propuesto:

- modelo en cascada
- modelo en espiral
- metodología ágil: scrum
- modelo iterativo e incremental (o creciente).

En relación a las metodologías anteriores, se han considerado las siguientes características para su evaluación:

- complejidad del modelo de desarrollo.
- complejidad del proyecto a construir.
- duración del proyecto.
- tipo de entregas.
- tipo de interacción con el cliente.
- dimensión del grupo de trabajo.
- experiencia previa en cada una de las metodologías.

En base a estos criterios se ha elegido el modelo iterativo e incremental, como la metodología que mejor se adapta a las características de este proyecto.

Desarrollo iterativo y creciente

El modelo de desarrollo iterativo e incremental surge como respuesta a las debilidades del modelo tradicional de cascada, del que se basa. El modelo en cascada sigue un enfoque

metodológico, en el que las distintas etapas del proceso de desarrollo de software se ordenan rigurosamente. De forma que cada etapa no puede comenzar hasta que no haya finalizado la etapa anterior.

En el modelo iterativo e incremental, se planifica un proyecto en distintos bloques temporales llamados iteración. En cada iteración se repiten las etapas del proceso de desarrollo software obteniéndose un producto funcional, que permite realimentar sucesivas iteraciones.

Las fases típicas en las que se estructura cada una de las iteraciones del proceso de desarrollo software son:

- Requisitos: obtención de los requisitos del sistema.
- Análisis: análisis de los requisitos, definición de alcance y límites del proyecto.
- Diseño: diseño de arquitectura y diseño detallado.
- Implementación o codificación.
- Pruebas: incluidas las pruebas funcionales, de sistema y de aceptación, para asegurar que el producto está construido según las especificaciones de los requisitos.
- Operación: Incluye la puesta en explotación del sistema, el mantenimiento y el soporte posterior del producto para garantizar que funcione correctamente en todo momento.

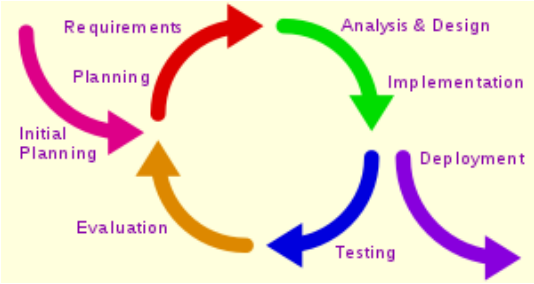


Figura 1 - Fases del modelo iterativo y creciente

Fuente:https://en.wikipedia.org/wiki/Iterative_and_incremental_development

1.4 Planificación

Se ha planificado la construcción del sistema en dos iteraciones del modelo iterativo e incremental, de forma que se minimice el riesgo ante imprevistos dadas las restricciones temporales para las distintas entregas y la finalización del trabajo.

En la primera iteración se ha obtenido un sistema que funcionalmente cumple con los principales requerimientos funcionales del sistema.

En la segunda iteración, se han añadido características adicionales, como mejoras en la disponibilidad de los componentes del sistema, principalmente del proveedor de servicios.

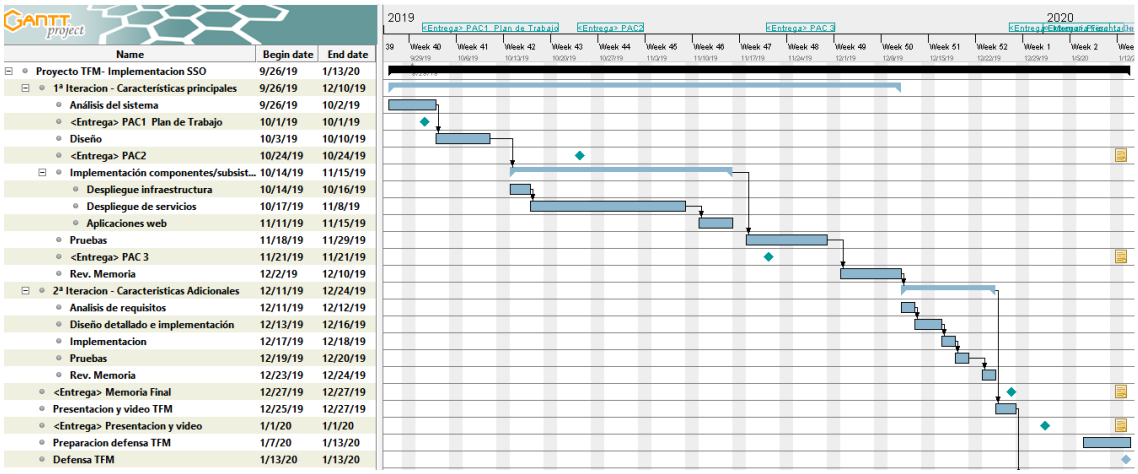
Planificación temporal

Se ha definido la siguiente lista de tareas a desarrollar a lo largo de este proyecto:

- Proyecto TFM- Implementacion SSO
  - 1ª Iteracion - Características principales
    - Análisis del sistema
    - <Entrega> PEC1 Plan de Trabajo
    - Diseño
    - <Entrega> PEC2
    - Implementación componentes/subsistemas
      - Despliegue infraestructura
      - Despliegue de servicios
      - Aplicaciones web
    - Pruebas
    - <Entrega> PEC 3
    - Rev. Memoria
  - 2ª Iteracion - Caracteristicas Adicionales
    - Analisis de requisitos
    - Diseño detallado e implementación
    - Implementacion
    - Pruebas
    - Rev. Memoria
  - <Entrega> Memoria Final
  - Presentacion y video TFM
  - <Entrega> Presentacion y video
  - Preparacion defensa TFM

Figura 2 - Lista de tareas definidas en la planificación del proyecto

La planificación temporal de las tareas indicadas anteriormente y sus dependencias se muestran en el siguiente diagrama de Gantt:



Name	Begin date	End date
Proyecto TFM- Implementacion SSO	9/26/19	1/13/20
1ª Iteracion - Características principales	9/26/19	12/10/19
Análisis del sistema	9/26/19	10/2/19
<Entrega> PAC1 Plan de Trabajo	10/1/19	10/1/19
Diseño	10/3/19	10/10/19
<Entrega> PAC2	10/24/19	10/24/19
Implementación componentes/subsist...	10/14/19	11/15/19
Despliegue infraestructura	10/14/19	10/16/19
Despliegue de servicios	10/17/19	11/8/19
Aplicaciones web	11/11/19	11/15/19
Pruebas	11/18/19	11/29/19
<Entrega> PAC 3	11/21/19	11/21/19
Rev. Memoria	12/2/19	12/10/19
2ª Iteracion - Caracteristicas Adicionales	12/11/19	12/24/19
Analisis de requisitos	12/11/19	12/12/19
Diseño detallado e implementación	12/13/19	12/16/19
Implementacion	12/17/19	12/18/19
Pruebas	12/19/19	12/20/19
Rev. Memoria	12/23/19	12/24/19
<Entrega> Memoria Final	12/27/19	12/27/19
Presentacion y video TFM	12/25/19	12/27/19
<Entrega> Presentacion y video	1/1/20	1/1/20
Preparacion defensa TFM	1/7/20	1/13/20
Defensa TFM	1/13/20	1/13/20

Figura 3 - Planificación temporal del proyecto

1.3 Organización de la memoria

La memoria de este trabajo está organizada en cuarto partes principales, junto a las referencias bibliográficas utilizadas y dos anexos, que complementa ciertos aspectos de la memoria.

En la introducción, se describe el problema que pretende resolver este trabajo, los objetivos que se han definido, la metodología utilizada y los principales aspectos de la planificación del proyecto. Por último, se describe la estructura de la memoria.

En la segunda parte se describe el estado del arte de las principales tecnologías que sirven de base para este proyecto. Se recogen los fundamentos teóricos de los sistemas de autenticación única (SSO) y se realiza una descripción del sistema de autenticación elegido.

En el tercera parte se aborda la implementación de un sistema de autenticación única y se recoge los detalles más relevantes de la construcción del mismo. En su último apartado, se realiza la validación del sistema a través del conjunto de pruebas, predefinidas en la fase de análisis.

En la cuarta parte se exponen las conclusiones que se han obtenido tras la realización de este trabajo. Además, se indican distintos aspectos de mejora o las líneas en las que se podría continuar con el trabajo desarrollado.

A continuación se incluyen las principales referencias bibliográficas que han sido utilizadas durante la construcción del sistema y preparación de esta memoria.

Por último, se añaden dos anexos que complementan distintos aspectos de la información dada en los bloques anteriores. En el anexo A se indican los principales requerimientos de los productos Shibboleth. Y en el anexo B, se muestran los detalles de la infraestructura de red desplegada.

2. Estado del arte

2.1 Autenticación/autorización

La autenticación es el proceso por el que se trata de verificar la identidad de una entidad, que puede ser un usuario, un proceso o un dispositivo. En cambio, la autorización es el proceso por el que se comprueba que la entidad que solicita acceder a un recurso, posee los permisos suficientes para acceder a él.

A nivel general, los sistemas de autenticación pueden clasificarse en distintas categorías según el elemento utilizado para verificar la identidad del usuario:

- Los que se basan en algo que se conoce, por ejemplo, en un contraseña o una frase de paso.
- Los que se basan en algo que se posee, por ejemplo, una tarjeta inteligente, un token de seguridad, …
- Los que se basan en una característica física del usuario o un acto involuntario del mismo, por ejemplo, en la verificación de voz, de escritura, de huellas dactilares,…

Por diferentes motivos, la mayoría de los sistemas de autenticación se basan principalmente en algo que se conoce, normalmente una contraseña. Si bien, cada vez se implantan más sistemas de doble o triple autenticación en los que, al menos, se utiliza otro elemento para realizar la verificación de la identidad del usuario.

2.2 Administración de identidades

La Administración de identidades está formada por el sistema integrado de políticas y procesos de una organización que pretende facilitar y gestionar el control de acceso a los sistemas de información. No se trata de un producto o una solución monolítica sino que representa un conjunto de soluciones relacionadas entre sí.

Son elementos de la administración de identidades:

- Gestión de Identidades: incluye la provisión de cuentas, la gestión de contraseñas,...
- Control de acceso: incluye las políticas de control de acceso y los sistemas de autenticación única (SSO), entre otros.
- Servicio de directorio: repositorio de información de los usuarios de la organización.

Dada la heterogeneidad de los escenarios en lo que pueden utilizarse estos sistemas son importantes los esfuerzos de estandarización que han surgido en este ámbito. A este respecto cabe destacar las siguientes iniciativas:

- Project Liberty, gestionado por el consorcio industrial Liberty Alliance.
- SAML de OASIS, consorcio internacional sin ánimo de lucro.
- Shibboleth, formado por un conjunto de estándares de identificación orientados a entornos educativos y administración pública.

Liberty Alliance donó su especificación de marco de trabajo sobre Federación de Identidades (ID-FF) a OASIS, constituyendo la base de SAMLv2. De forma que SAMLv2 representa la convergencia los principales estándares existentes, SAML v1.1, Liberty ID-FF 1.2 y Shibboleth 1.3.

2.3 SSO

Los sistemas de autenticación única (SSO) son procedimientos de autenticación que permiten a un usuario acceder a varios sistemas realizando un único proceso de autenticación.

En un sistema de autenticación única podemos identificar los siguientes elementos:

- un conjunto de recursos o aplicaciones a los que acceder de forma controlada.
- un conjunto de usuarios que desean acceder a dichos recursos.
- un proveedor de servicios que gestiona el acceso de los usuarios a los recursos utilizando los servicios de autenticación de un proveedor de identidad.
- un proveedor de identidad que verifica la identidad de los usuarios.

Ventajas/inconvenientes de los SSO

Las principales ventajas que podemos encontrar en el uso de sistemas SSO:

- disminuye la fatiga de autenticación o de contraseña.
- aumenta la usabilidad del sistema al disminuir el número de ocasiones en las que se ha de verificar la identidad del usuario.
- la división de funciones que suele darse en estos facilita la implementación de cada uno de los componentes (mecanismos de autenticación, consulta de atributos,…).
- simplifica la integración de información de usuario de distintas fuentes, ya sea bases de datos, servicios de directorio, etc.
- disminuye el riesgo de caer en conductas que reduzcan la seguridad del sistema.

A pesar de las ventajas anteriores, también aparecen algunos inconvenientes o consideraciones a tener en cuenta, en el uso de estos sistemas:

- En caso de ser comprometida la cuenta de un usuario, repercute en todos las aplicaciones integradas en el SSO.
- El sistema de autenticación se convierte en un sistema crítico.
- Aumenta la complejidad del sistema.
- Se basa principalmente en la protección de las contraseñas, que no es el mecanismo más fiable. No olvidemos que frecuentemente los usuarios son eslabón más débil del sistema de autenticación. Si bien cada vez más, se implementan sistemas con sistemas alternativos no basados solo en contraseñas.
- Los sistemas SSO proporcionan una solución a la autenticación de los usuarios. Sin embargo, no ofrecen una solución completa a los distintos niveles de control de acceso que se requieren en las aplicaciones empresariales. En las aplicaciones normalmente es necesaria una mayor granularidad en el control del acceso a los recursos, por lo que suele ser necesario implementar mecanismos de autorización adicionales.
- La independencia de los mecanismos de autenticación únicos puede verse comprometida por las políticas comerciales de las empresas que los implementan. Por ejemplo, el bloqueo de cuentas de redes sociales en países en los que existe censura en el acceso a ciertos contenidos, puede de facto impedir al usuario el uso de terceras aplicaciones que no aplican, ni probablemente estén de acuerdo, con este bloqueo.

Implementaciones SSO

Hay múltiples implementaciones de los sistemas de autenticación únicas, cada cual con distintas características específicas. En base a las características del proyecto que se pretende implementar y a la popularidad de las distintas soluciones del mercado, se han evaluado las siguientes soluciones Web SSO:

- **CAS**: implementación Web SSO que soporta distintos protocolos para el proceso de autenticación, por ejemplo CAS, SAML v1, SAML v2, OAuth2, SCIM, OpenID Connect y protocolos WS-Federación.
- **Shibboleth**: Implementación SSO basado en el protocolo SAML v2 que permite el uso de distintos mecanismos de autenticación como CAS. Permite el uso de una identidad en distintos sistemas federados de distintas organizaciones.

Ambas implementaciones soportan múltiples protocolos de autenticación, su funcionalidad es muy similar y están muy extendidas, cada una en su ámbito. Se ha elegido la implementación de Shibboleth en base a las siguientes características:

- Gestión de atributos. Una de los elementos diferenciadores de la implementación de Shibboleth es la flexibilidad y la potencia de sus mecanismos para recuperar información asociada al usuario del repositorio de la organización.
- Gestión de identidad federada. Esta permite la interrelación de distintos proveedores de identidad y de servidores entre sí, de forma que no han de pertenecer a la misma organización. Para lo que se basa en el uso de protocolos estándares como SAML v2.
- Servicio descubrimiento. Shibboleth permite poner en marcha un servicio de descubrimiento dinámico que permite a los proveedores de servicios localizar los distintos proveedores de identidad que están accesibles (misma federación) en lugar de tener que estar vinculados estáticamente.

2.4 SAML v2

Security Assertion Markup Language (SAML) es un protocolo estándar para el intercambio de información de autenticación y autorización entre dominios de seguridad. Está basado en XML y define como se transmiten tokens de seguridad, con información de un principal, entre un proveedor de identidad y un proveedor de servicios.

Los principales escenarios en lo que se puede utilizarse SAML incluye Web SSO, identidad federada o servicios webs. En cada uno de ellos aporta una serie de ventajas. Centrándonos en el entorno web, la mayoría de las implementaciones de SSO tradicionales utilizan cookies en el navegador para mantener la información de estado. Sin embargo, las cookies no puede transmitirse entre dominios DNS, por lo que no posible utilizar este mecanismo entre sistemas con distintos dominios. Existen soluciones propietarias que dan soporte a SSO multidominio. Sin embargo, debido a la gran heterogeneidad de los entornos de las aplicaciones empresariales, el uso de soluciones propietarias interoperables resulta muy complicado. SAML soluciona este problema proporcionando un estándar que permite la implementación de soluciones multidominio independientes del proveedor.

Participantes

En una comunicación SAML interviene distintas entidades, principalmente una entidad que genera aserciones (SAML asserting party) y otras entidades que utilizan las aserciones (SAML relying party).

Las entidades SAML puede funcionar con distintos roles, según las funciones que desempeñen. Por ejemplo, en un SSO multidominio (MDSSO o SSO), SAML define los roles de proveedor de identidad y proveedor de servicios. Respecto a la gestión de atributos de un usuario, se define el rol de autoridad de atributos (attribute authority) que responde a las consultas de solicitantes de atributos (attribute requesters).

El elemento más importante de la mayor parte de las aserciones SAML es un objeto *principal*, que es la entidad que trata de ser autenticada dentro de un contexto de un dominio de seguridad. Puede tratarse den una persona u otro tipo de entidad como un proceso o dispositivo.

Componentes

La siguiente figura muestra la relación entre los conceptos básicos de SAML:

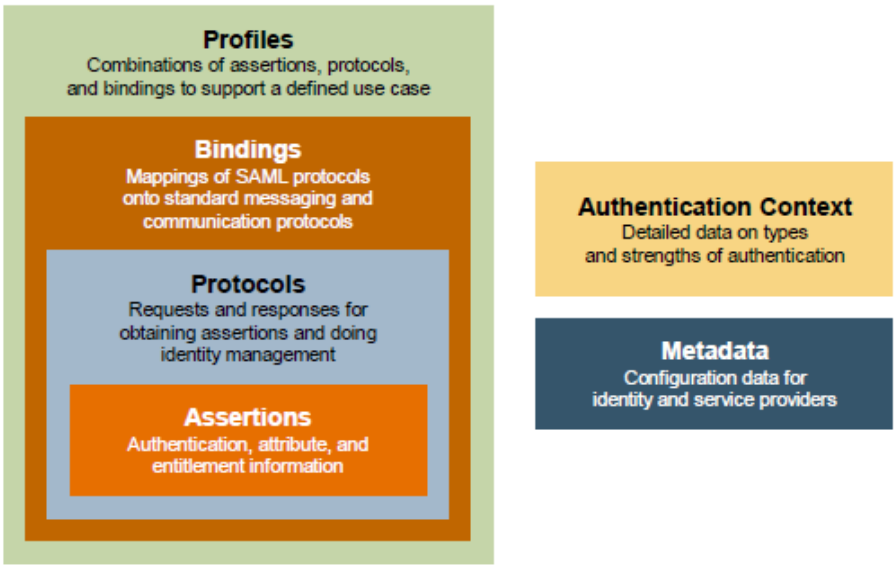


Figura 4 - Conceptos básicos de SAML. Fuente: SAML V2.0 Technical Overview

Aserciones

Las aserciones son declaraciones que hacen ciertas entidades SAML que son utilizadas por otras entidades SAML. Se definen tres tipos de declaraciones o sentencias:

- de autenticación: Son generadas por la entidad que autentica al usuario. Indican, al menos, el mecanismo de autenticación utilizado y la fecha en que se ha producido la misma.
- de atributos: Contienen atributos relativos a un principal..
- de decisión de autorización: que informan si se acepta o deniega el acceso de un usuario a un recurso protegido.

Protocolos

SAML define distintos protocolos de petición/respuesta:

- **Petición de autenticación**: define como un principal (o alguien en su nombre) puede solicitar su autenticación (peticiones de sentencias de autenticación) u obtener información del usuario (petición de atributos).
- **Single Logout**: Define un mecanismo que permita realizar un logout "casi simultaneo" de las distintas sesiones activas que tenga un principal.
- **De petición/consulta**: define un conjunto de consultas o peticiones de las entidades SAML a través de distintas aserciones.

Bindings

Los bindings definen como los mensajes de los distintos protocolos SAML puede ser transmitidos sobre los distintos protocolos de transporte, SOAP y HTTP principalmente. Los bindings definidos en SAML V2.0 son:

- **Redirección HTTP**: Define como mensajes del protocolo SAML puede ser transportados usando mensajes de redirección HTTP (código HTTP 302).
- **Peticiones POST HTTP**: Define como se transmiten mensajes SAML utilizando campos de control de formularios HTML, codificados en base64, mediante peticiones POST.
- **Artefactos HTTP**: Define como transmitir un artefacto usando HTTP, bien a través de controles de formularios HTML o de la propia URL (query string).
- **SAML SOAP**: Define como transmitir mensajes SAML a través de mensajes SOAP 1.1 (SOAP sobre HTTP).
- **SOAP inverso (PAOS)**: Define el intercambio de mensajes que permite a un cliente HTTP responder mensajes SOAP. Es utilizado en los perfiles de cliente mejorado (Enhanced Client) y Proxy (Proxy Profile). Diseñado para gateways WAP.
- **SAML URI**: Define como extraer una aserción SAML resolviendo un identificador de recurso URI (uniform resource identifier).

Perfiles (profiles)

Los perfiles definen como colaboran los distintos elementos (aserciones, protocolos y bindings) en un escenario de uso concreto. Los principales perfiles definidos en SAMLv2 son:

- **Web Browser SSO**: Define como las entidades usan el protocolo petición de autenticación junto con distintas aserciones para implementar el mecanismo de single sing-on en navegadores web. Define como se utilizan distintos mensajes junto a redirecciones HTTP, peticiones POST HTTP y artefactos.
- **Descubrimiento del proveedor de identidad**: Define un mecanismo por el que los proveedores de servicio pueden tener información sobre los proveedores de identidad que un usuario ha visitado previamente.
- **Logout simple** (single logout**)**: Define como el protocolo Single Logout puede implementarse mediante mensajes SOAP, redirecciones HTTP, peticiones POST HTTP o artefactos HTTP.
- **Cliente mejorado /Proxy** (Enhanced Client and Proxy - ECP): Define clientes "mejorados" (no navegadores web) y gateways pueden utilizar bindings PAOS y SOAP.

Otros perfiles definidos son el de resolución de artefactos, el de peticiones/respuesta, el de gestión de identificadores de nombre o el de mapeo de identificadores de nombres.

Perfil Web Browser SSO –redirección HTTP/POST HTTP

En este apartado se describe el flujo típico de mensajes que tiene lugar en el perfil Web browser SSO de SAML v2.0. Este perfil define el uso de mensajes SAML y binding que dan soporte al caso de uso Web SSO.

Respecto a quien comienza el proceso de autenticación, pueden darse dos variantes:

- proveedor de identidad: el usuario visita el proveedor de identidad, en el que se produce la autenticación. Posteriormente, selecciona un enlace que le lleva al proveedor de servicios donde se encuentran los recursos a los que va a acceder.
- proveedor de servicio: el usuario intenta acceder a un recurso ofrecido por un proveedor de servicios, que comprueba si es necesario autenticar al usuario.

La solución que se plantea en este trabajo se ajusta a este segundo escenario. En la siguiente figura se observan los distintos mensajes que tiene lugar para completar la petición del usuario.

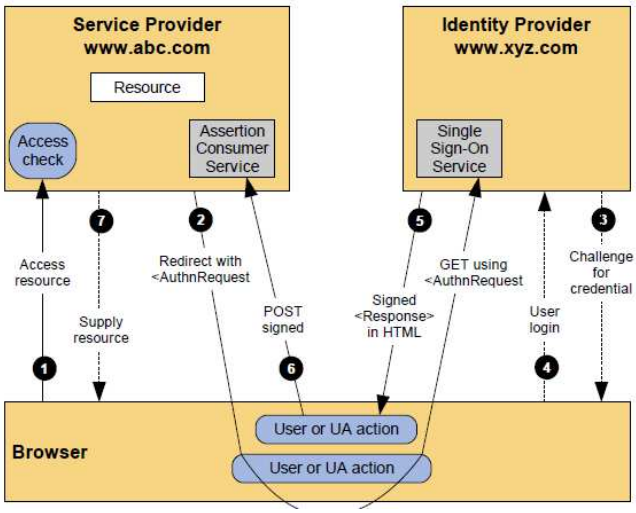


Figura 5 - Flujo en perfil Web SSO iniciada por SP. Fuente: SAML V2.0 Technical Overview

1. El usuario intenta acceder al recurso en el proveedor de servicio (SP). Suponemos que el usuario no tiene una sesión válida en este sitio. El SP almacena la URL solicitada como información local que puede ser transmitida.

2. El SP envía una respuesta de redirección HTTP al navegador (código HTTP 302 o 303). La cabecera Location contiene la dirección URI del servicio de autenticación SSO en el proveedor de identidad junto con un mensaje <AuthnRequest> codificado como la variable del query string SAMLRequest. El navegador procesa la redirección y hace una petición GET a la URL indicada en el parámetro SAMLRequest. La información de estado es también incluida en la respuesta, a través de un parámetro RelayState del query string.

3. El servicio de autenticación SSO comprueba si el usuario tiene un contexto de seguridad en el proveedor de identidad que cumpla los requisitos indicados en el mensaje <AuthnRequest>. Si no es así, el proveedor de identidad (IdP) interactúa con el navegador para solicitar las credenciales al usuario.

4. El usuario proporciona unas credenciales válidas y se crea un contexto de seguridad para el usuario en el IdP.

5. El servicio SSO construye una aserción representando el contexto de seguridad. Se utiliza como binding una petición POST, de forma que la aserción es firmada e incrustada en un mensaje de respuesta SAML. Que a su vez es incrustada en un campo de control oculto del formulario HTML llamado SAMLResponse. Si el IdP ha recibido un valor RelayState desde el SP, lo reenvía sin modificar en un control oculto llamado RelayState. El servicio SSO envía el formulario de vuelta al navegador en la respuesta HTTP. Normalmente para facilitar el proceso de reenvío, el formulario HTML se acompañará de un script que automáticamente reenviará el formulario al sitio de destino.

6. El navegador, bien debido a la ejecución del script de autosubmit o a la acción del usuario, envía una petición POST al servicio de gestión de aserciones del SP. Este servicio obtiene el mensaje de respuesta procesando los campos del formulario HTML. Se comprueba la validez de la aserción (incluida la firma). Y se procesa el contenido de la aserción, dando lugar a la creación de un contexto local de seguridad para el usuario en el SP. Una vez finalizado el procesamiento, el SP recupera la información de estado local, incluida la URL del recurso inicialmente solicitado (RelayState), para invocar este recurso. Se envía una respuesta de redirección al navegador indicando la URL del recurso solicitado.

7. Puede producirse una comprobación de acceso adicional de forma opcional, encargada de verificar si el usuario tiene autorización para acceder al recurso. En caso afirmativo, el recurso es devuelto al navegador, de forma que el usuario puede tener acceso al mismo.

2.5 Shibboleth

La arquitectura de Shibboleth define un conjunto de interacciones entre un proveedor de identidad y un proveedor de servicio para facilitar la autenticación de usuarios en aplicaciones web junto al intercambio de atributos.

Los principales objetivos de Shibboleth son:

- Permitir a una organización usar mecanismos de autenticación para el acceso a recursos web aunque el recurso no sea gestionado por la organización.
- Permitir que la información de usuario gestionada por la organización pueda ser transferida al recurso o aplicación.
- Permitir al usuario, o su organización, controlar la cesión de la información.

Si bien la mayor parte de la información que se incluye sobre los productos de Shibboleth es aplicable a las distintas versiones, para la realización del trabajo se han utilizado:

- Proveedor de identidad: Shibboleth IdP v3
- Proveedor de servicio: Shibboleth SP v3

Las entidades que aparecen en el esquema de Shibboleth son:

- **Proveedor de identidad** (IdP): es una entidad que autentica a un principal, produce aserciones de autenticación y atributos de información.
- **Proveedor de servicio** (SP): es una entidad que controla el acceso a un servicio, aplicación o recurso web. Para realizar sus funciones procesa o consume aserciones generadas, normalmente, por un proveedor de identidad. También puede incluir la funcionalidad de solicitante de atributos.

Proveedor de identidad

El proveedor de identidad de Shibboleth (Shibboleth IdP) implementa las siguientes funciones o roles del modelo de dominio SAML:

- **autoridad de autenticación**: es un servicio definido en SAML que genera aserciones de autenticación sobre un principal para otras entidades (relaying parties) como los proveedores de servicios.
- **autoridad de atributos**: es un servicio que procesa peticiones de atributos.
- **servicio single sing-on**: es un recurso HTTP, controlado por el proveedor de identidad, que recibe y procesa las peticiones de autenticación del proveedor de servicios.
- **servicio de transferencia inter sitio** (inter-site transfer service): es un recurso HTTP, controlado por el proveedor de identidad, que interactúa con la autoridad de autenticación para generar respuestas al navegador del usuario.

En la figura se muestra los elementos en la arquitectura del proveedor de identidad Shibboleth. Las peticiones HTTP son atendidas por el gestor de peticiones (Request Dispatcher). Este inspecciona la petición y la envía a un gestor de perfiles. El gestor de perfiles gestiona las peticiones asociadas a un determinado perfil (p. ej. SAML v1 Consulta de atributos, SAML v2 SSO).

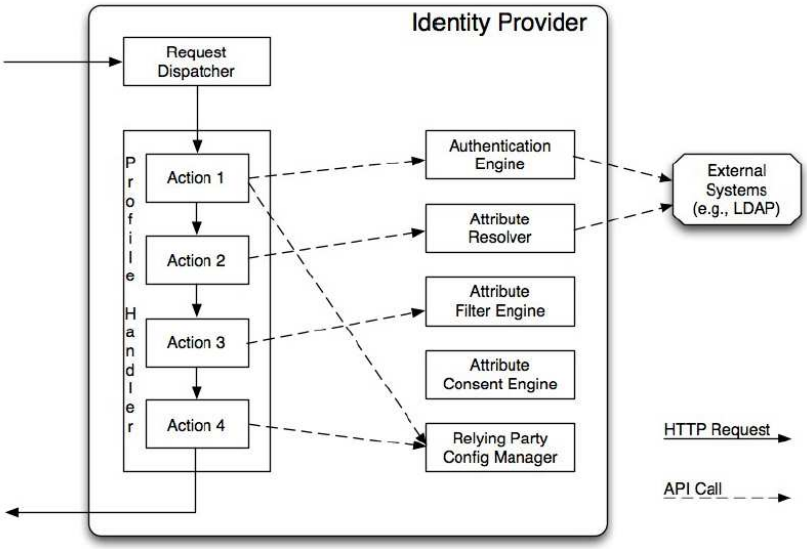


Figura 6 - Arquitectura de proveedor de identidad de Shibboleth. Fuente: https://wiki.shibboleth.net/confluence/display/IDP30/GeneralArchitecture

En Shibboleth, las funciones de gestor de peticiones y de perfiles son implementadas mediante la definición de flujos declarativos de altos nivel utilizando Spring Web Flow. Cada flujo está formado por un conjunto de acciones en las se realiza parte del procesamiento para generar la respuesta apropiada a la petición.

Autenticación

Los ficheros de la carpeta *conf/authn/* están relacionados con los mecanismos de autenticación definidos en Shibboleth IdP. Respecto a la configuración general, en el fichero *authn/general-authn.xml* en el bean llamado *shibboleth.AvailableAuthenticationFlows* se describen los mecanismos de autenticación que pueden ser utilizados por el IdP. El identificador del flujo se corresponde con el nombre del fichero de definición del flujo en la carpeta *flows/authn/.*

Para que un flujo pueda ser utilizado por el IdP, no es suficiente estar definido en el bean. También ha de estar habilitado, para lo cual su identificador ha de estar definido en la propiedad *idp.authn.flows*, ya que en otro caso será ignorado.

Los mecanismos de autenticación (login flows) proporcionados por Shibboleth IdP3 son:

- Password
- X509
- Multi-Factor
- Otros mecanismos: RemoteUser, RemoteUserInternal, X509Internal, SPNEGO / Kerberos, IPAddress, External, Duo.

Otras propiedades interesantes definidas en este fichero son:

- *idp.session.enabled*: permite des/habilitar la funcionalidad de seguimiento de sesión. Si se deshabilita se inhibe la funcionalidad SSO del proveedor de identidad de forma que se autentica cada petición de forma independiente.
- *idp.session.StorageService:* indica el nombre del bean que gestiona el almacenamiento de las sesiones del IdP, por defecto, su valor es *shibboleth.ClientSessionStorageService*.

Atributos

La entidad encargada de resolver atributos (Attribute Resolver) realiza tres tareas: extrae los datos de un sistema externo (p.ej. servicio de directorio o base de datos relacional), crea los atributos de la información obtenida y asocia un decodificador a los atributos generados.

Un atributo, en un IdP, es una estructura de datos que contiene un identificador único, un conjunto de nombres, de descripciones, de valores para dicho atributo y de codificadores, que puede transformarlo en una representación específica de un protocolo.

Las entidades que resuelven atributos están formadas por tres componentes: conectores de datos, definiciones de atributos y codificadores de atributos.

- **Conectores de datos**: permiten extraer información de fuentes externas como servicios de directorio o bases de datos relacionales.
- **Definición de atributos**: en las que se define cada atributo, indicando sus codificadores, nombres y descripciones.
- **Codificaciones de atributos**: encargados de generar para cada atributo la estructura XML requerida por SAML2.
- **Decodificadores de atributos**: inspeccionan los metadatos (normalmente del SP) para extraer de estos la información de los atributos (RequestedAttributes).

Clustering

Shibboleth IdP es una aplicación con estado. Entre distintas peticiones de un usuario, mantiene cierta información de estado, necesaria para procesar las siguientes peticiones.

En Shibboleth IdP se mantienen distintos tipos de información de estado:

- estado conversacional de los flujos de Spring Web Flow durante el procesamiento de la petición de un perfil (peticiones de login, consulta de atributos,…)
- sesiones del IdP manteniendo los resultados de la autenticación, de forma que puedan ser reutilizados en sucesivas peticiones de autenticación y/o por los servicios de seguimiento de logout (conocer que peticiones hay activas para poder cerrarlas).
- información sobre las distintas opciones sobre la autorización para la cesión de información de usuario (si se pregunta en cada petición, cuando cambie el proveedor de servicios, …)
- cache de mensajes enviados: usada para almacenar los identificadores de los mensajes enviados para prevenir ataques de repetición.
- almacenamiento de artefactos SAML.

Respecto al estado conversacional, el IdP necesita que los flujos se completen en el mismo nodo en que se ha iniciado. Debido a que la información de sesión del IdP no es serializable (en el contexto de java), no puede hacerse persistente y ser transferidas de un nodo a nodo de un clúster. Esta decisión simplifica el diseño del IdP pero dificulta su escalabilidad. En definitiva, no hay solución para replicar el estado conversacional de cada petición por lo que soluciones cien por cien de alta disponibilidad no son posibles. Es necesario mantener algún grado de afinidad de las peticiones con los nodos en los que se trata la petición (stickiness).

Proveedor de servicio

El proveedor de servicio intercepta las peticiones de acceso a los recursos de las aplicaciones, comprueba los requisitos de acceso y, si es necesario, las redirige al proveedor de identidad.

El proveedor de servicio Shibboleth SP está formado por dos elementos:

- servidor web.
- shibd: implementa la funcionalidad específica del proveedor de servicios. Este componente se implementa como un servicio en Windows y un demonio en Linux/Unix.

La configuración se realiza a través de los ficheros en la carpeta etc/shibboleht y la configuración del servidor web. En el caso de Linux, a través de un módulo de apache.

Uno de los ficheros de configuración más importantes es shibboleht2.xml, en el que se define el elemento ApplicationDefault. Este elemento define la mayor parte del comportamiento en lo referido a SAML y las políticas de sesión:

- entityID: permite definir el identificador de identidad SAML (SAML entityID) utilizado por el proveedor de servicio.
- SSO: elemento que da soporte a protocolos de autenticación única.
- CredentialResolver: elemento que referencia a las claves de descifrado y firma para las comunicaciones SAML.
- metadatos del proveedor de servicio: a través de un elemento <MetadataProvider>.
- REMOTE_USER: variable de servidor que referencia la identidad principal del usuario.

Gestión de atributos

Por defecto, Shibboleth SP ignora la información de los usuarios (atributos adicionales) que llega a través de aserciones SAM. Para indicarle explícitamente que las procese, es necesario definir una serie de reglas que permitan asociar los atributos de las aserciones (propiedad nombre de la regla) a un identificador con el que el SP podrá referenciarlas (propiedad id de la regla). Con el nombre indicado en la regla se creará una variable de entorno o cabecera HTTP a través de la cual se expondrá esta información a la aplicación web. Estas reglas se definen en el fichero *attribute-map.xml*.

A los aquellos atributos cuyos valores tengan cierta estructura, no sean solo cadenas de texto, es posible asociarles un decodificador que indicará como extraerá el valor de forma adecuada.

Clustering

Shibboleth SP mantiene cierta información de estado en sus componentes, en el servicio web y en el demonio, que dificulta la compartición de estado entre nodos de un clúster. Debido a esto, es necesario mantener cierta afinidad entre las peticiones del usuario y el nodo que procesa la petición, el tiempo necesario para completar el proceso de autenticación. Una vez se finalice la comunicación con el IdP (recepción de aserciones) y se reenvíe la petición, el resto del tráfico puede ser procesado por cualquiera de los nodos.

La mayoría de las aplicaciones tienen sus propios mecanismos de sesión, de forma independiente a la sesión del SP.

2.6 Alta disponibilidad

Dos características importantes en la evaluación de un sistema informático son:

- **Escalabilidad**: capacidad de un sistema de hacer frente a cargas de trabajos crecientes de manera eficiente, prestando un nivel de rendimiento aceptable.
- **Disponibilidad**: medida del grado en que un sistema es utilizable, es decir, está preparado para recibir carga de trabajo.

Relacionados con estos dos características de un sistema, está el concepto de alta disponibilidad. Una configuración de alta disponibilidad es aquella que permite asegurar que el tiempo que el sistema no está disponible es mínimo, normalmente por debajo de un valor mínimo comprometido.

Un elemento esencial para alcanzar alta disponibilidad es la eliminación de los puntos únicos de fallo. Un punto único de fallo es aquel elemento del sistema cuyo fallo provoca que el sistema no sea utilizable con un rendimiento aceptable. Una de las estrategias más utilizadas para conseguirlo consiste en agregar redundancia a cada capa de la arquitectura del sistema.

Por otra parte, es necesario tener en cuenta que, cuanto más complejo sea la arquitectura de un sistema, es probable que tenga más puntos de fallo, por lo que conseguir alta disponibilidad puede ser más complicado.

Escalabilidad

Podemos utilizar dos estrategias para aumentar la escalabilidad de un sistema:

- **escalado vertical**: consiste en utilizar un hardware más potente y eficaz que el actual. Se consigue una mayor prestación asignando más recursos, más procesadores, más memoria, más almacenamiento…
- **escalado horizontal**: consiste en aumentar el número de nodos que prestan el servicio.

En general, la estrategia de escalado horizontal aporta una serie de ventajas:

- suele ser más económico, ya que normalmente el coste total de un sistema de mayor capacidad de procesamiento es mayor que el de un conjunto de n sistemas de capacidad total equivalente.
- ofrece redundancia al disponer de más de un nodo prestando el mismo servicio.
- mayor escalabilidad, normalmente la capacidad de ampliación de recursos a un sistema único está más limitada que la de añadir nodos.

No todo son ventajas en el escalado horizontal. Entre las principales desventajas, suele requerir un rediseño del sistema, y normalmente un aumento de la complejidad del mismo.

Balanceo de carga

El balanceo de carga es una técnica que consiste en la agregación de múltiples componentes para alcanzar una capacidad de procesamiento total superior a la de cada uno de los componentes de forma individual, sin intervención del usuario final.

Requiere de la existencia de un componente llamado balanceador de carga que recibe las peticiones y las distribuye, de acuerdo a un algoritmo de reparto, entre los distintos componentes. Este componente puede ser software o hardware.

Los algoritmos de balanceo de carga determinan como se realiza el reparto de las peticiones entre los servidores. Algunos de los algoritmos utilizados más frecuentemente son:

- **Round robin**: se distribuye la carga en forma equitativa entre los servidores.
- **Least connections** (LC): se selecciona el servidor con el menor número de conexiones.
- **Source**: se selecciona el servidor según el resultado de aplicar una función hash a la dirección IP del usuario.

Puede ser necesario para algunas aplicaciones que una vez un usuario se conecte a un servidor, al menos durante un tiempo, las sucesivas peticiones sean atendidas por el mismo servidor. Es lo que se conoce como afinidad o apego al servidor (stickiness). Al llegar la primera petición de un usuario se asigna según el algoritmo de balanceo un servidor y se crea una asociación (sticky session) que estará vigente por cierto tiempo. Mientras esté activa, el resto de peticiones que realice dicho usuario, se enviarán al mismo nodo o servidor.

HAProxy

HAProxy (High Availability Proxy) es un software de código abierto que proporciona funcionalidad de balanceo de alta disponibilidad y de servidor proxy para TCP y HTTP. Está escrito en C y es bastante rápido y eficiente en términos de uso de procesador y memoria.
En HAProxy se definen tres elementos principales: frontend, backend y listas de acceso (ACLs).

Un frontend permite definir un punto de entrada de peticiones de usuario. Los principales elementos de su configuración son:

- Punto de entrada de las peticiones, indicando direcciones IP y puerto, p.ej. 10.1.1.7:80.
- Backend o conjuntos de servidores a los que se reenvía la petición.
- Definición de listas de acceso o ACLs.

Un ejemplo de definición de un frontend que procesa peticiones https:

```
frontend nombre_frontend
  bind *:443
  use_backend nombre_backend
```

Un backend está formado por un conjunto de servidores que procesan un mismo tipo de peticiones del usuario. En la configuración de un backend, se define el algoritmo de balanceo de carga y los servidores a los que se reenvían las peticiones.
Se muestra un ejemplo de definición de backend formado por dos servidores:

```
backend nombre_backend
  balance roundrobin
  server servidor1:443 check
  server servidor2:443 check
```

Las listas de control de acceso (ACL) son un elemento importante en la configuración de HAProxy. Permiten definir conjuntos de elementos que cumplen una condición para posteriormente pode ejecutar acciones en base a estos. Por ejemplo, es posible definir una ACL para las peticiones cuya URL cumpla cierta condición y luego realizar ciertas acciones sobre dichas peticiones. Se muestra como ejemplo, como se puede usar una ACL para que las peticiones cuya URL que comiencen por /estatico sean asignadas a un backend en concreto:

```
acl es_estatico path -i -m beg /estatico
use_backend bk_estatico if es_estatico
```

Como se ha indicado HAProxy actúa como proxy inverso, es decir, como intermediario de las peticiones que los usuarios hacen a los servidores que proporcionan un servicio. Los proxys inversos permiten mantener oculta la estructura de la subred interna en la que se despliegan los servicios y facilitan las configuraciones de balanceo d carga.

Si nos centramos en peticiones web utilizando HTTPS HAProxy permite dos modos de funcionamiento:

- **SSL Termination**. La conexión SSL del cliente acaba en el balanceador. Supone una mayor carga para el balanceador que ha de realizar el des/cifrado de la información. La conexión desde el balanceador hasta el servidor se realiza normalmente en claro, sin cifrar.
- **SSL Pass-through**. El balanceador solo redirige las peticiones a los servidores. Dado que la información está cifrada no es posible procesar la petición en base a la información que contiene ni modificarla. Es más seguro ya la conexión se mantiene cifrada hasta el servidor que procesa la petición.

La configuración de los distintos elementos de HAProxy se realiza a través del fichero /etc/haproxy/haproxy.cfg. Además de las secciones correspondientes a los elementos mencionados, contiene una sección global (o defaults) en la que se definen los parámetros generales de funcionamiento HAProxy.

3. Solución propuesta

3.1 Análisis de requisitos

El análisis de requisitos tiene como objetivo “el estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, de hardware o de software, así como el proceso de estudio y refinamiento de dichos requisitos” (Estándar IEEE Std. 610 [IEEE 1990]).

Se muestran los distintos requisitos que se han obtenido en las distintas iteraciones del proyecto.

Requisitos funcionales

Los requerimientos funcionales, son declaraciones de las funciones que proporcionará el sistema. Se han obtenido los siguientes requerimientos funcionales:

RF_SSO_01: Las aplicaciones integradas en el sistema tienen una parte pública y otra privada, que está protegida. Para acceder a la parte privada el usuario tiene que estar autenticado.

RF_SSO_02: El sistema ha de permitir a un conjunto de aplicaciones web compartir los mecanismos de autenticación para el acceso a sus recursos protegidos.

RF_SSO_03: El sistema ha de permitir la autenticación mediante usuario y contraseña.

RF_SSO_04: El sistema ha de permitir autenticar al usuario mediante certificado electrónico.

RF_SSO_05: Si el usuario ha sido autenticado por una aplicación del sistema, al menos durante cierto tiempo, podrá acceder a los recursos de otras aplicaciones sin tener que autenticarse.

RF_SSO_06: Si un usuario autenticado sale de una aplicación (hace logout), al volver a acceder a cualquier recurso protegido, será necesario comprobar su identidad.

RF_SSO_07: Un usuario autenticado, al acceder a un recurso o aplicación, expondrá cierta información obtenida del servicio de directorio. Al menos será accesible el nombre completo, el email, el tipo de empleado y su posición o función en la empresa.

RF_APP_08: El usuario puede salir de una aplicación y dejar de estar autenticado desde cualquier página de la parte protegida. Para ello existirá un enlace “Salir de la aplicación” u otro elemento de la interfaz gráfica de propósito similar.

Requisitos no funcionales

Los requerimientos no funcionales recogen otros aspectos el sistema, relacionados con la operación del sistema, seguridad, aspectos legales, etc.

Requisitos operativos

RNF_SSO_09: La información de autenticación de los usuarios del sistema SSO se almacenará en servicio de directorio activo.

RNF_SSO_10: Un proveedor de identidad que se encarga de verificar la identidad de los usuarios de las aplicaciones.

RNF_SSO_11: Las aplicaciones utilizaran los servicios de autenticación ofrecidos el sistema para realizar la autenticación de los usuarios, como paso previo a permitirles el acceso a su sección privada.

RNF_SSO_12: Las aplicaciones y el mecanismo de autenticación mantendrán la mayor compatibilidad posible con navegadores web. Al menos será compatible con las últimas versiones de los principales navegadores web.

RNF_APP_13: La interfaz de usuario de las aplicaciones será implementada para navegadores web compatibles con HTML5 y JavaScript.

RNF_SSO_14: La comunicación entre el proveedor de servicios y el proveedor de identidad se podrá realizar utilizando distintos protocolos. Como mínimo utilizará el estándar SAML2.

Seguridad

RNF_SSO_15: Las comunicaciones entre el usuario y el sistema de autenticación estarán cifradas utilizando SSL o protocolo de similar nivel de protección.

RNF_SSO_16: Las comunicaciones entre el sistema SSO y el servicio de directorio estarán cifradas utilizando SSL o protocolo de similar nivel de protección.

RNF_SSO_17: Las contraseñas de usuarios en el servicio de directorio se almacenaran de forma segura utilizando alguna función resumen basada en SHA u otra con igual nivel de protección o superior.

Requisitos legales

RNF_SSO_20: Utilización de productos de software libre para la implementación del sistema.

Requisitos económicos

RNF_SSO_21: El proyecto se plantea en el marco de un Proyecto Fin de Máster. El coste económico debe ser mínimo, por lo cual, es recomendable:

- Uso de componentes software de acceso libre.
- Implementación en un entorno virtualizado.

3.2 Análisis

Casos de uso

A continuación se muestra el diagrama de caso de uso del sistema, en el que se representan los principales requerimientos funcionales:

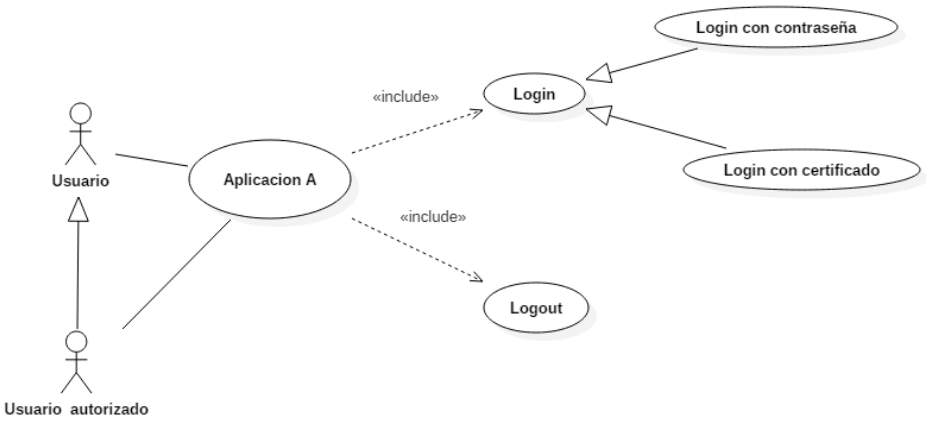


Figura 7 - Diagrama de Caso de Uso CU01

Los actores que aparecen en el caso de uso son:

- **Usuario**: accede desde un navegador a los recursos públicos de las aplicaciones.
- **Usuario autorizado**: está autorizado a acceder a los recursos protegidos de las aplicaciones. Sus datos están en el servicio de directorio de la organización.

A continuación se describen los escenarios más frecuentes del caso de uso del sistema.

CU01-e01 Acceso a un recurso público de la aplicación A

CU01-e01	Acceso a recurso publico	
Actor:	Usuario	
Descripción:	Procedimiento para el acceso a recurso público de la aplicación A	
Precondiciones:	El usuario no tiene por qué estar en el servicio de directorio El usuario no está autenticado	
Poscondiciones:		
Flujo normal:	1	El usuario escribe la URL del recurso público de la aplicación
	2	El usuario accede al recurso de la aplicación

Tabla 1 - Descripción del escenario e01 del caso de uso CU01

CU01-e02 Acceso a recurso protegido con contraseña (Login)

CU01-e02	Acceso a recurso protegido de la aplicación con contraseña		
Actor:	Usuario autorizado		
Descripción:	Procedimiento para el acceso a un recurso protegido de una aplicación utilizando como información de autenticación el identificador de usuario y contraseña		
Precondiciones:	El usuario no está autenticado. El usuario ha autorizado la cesión de datos al proveedor de servicios.		
Poscondiciones:	El usuario está autenticado en el sistema o recibe un mensaje de error.		
Flujo normal:	1	El usuario intenta acceder a un recurso protegido de la aplicación	
	2	Se presenta al usuario un formulario para que introduzca el usuario y contraseña	
	3	El usuario introduce el usuario y contraseña	
	4	El sistema comprueba los datos enviados por el usuario.	
	5	a	Los son correctos, se autentica al usuario y se accede a su información adicional (nombre completo, tipo empleado, …)
Flujos alternativo:	5	b	Los datos enviados no son correctos y se muestra al usuario un mensaje de error

Tabla 2 - Descripción del escenario e02 del caso de uso CU01

CU01-e03 Acceso a recurso protegido con certificado electrónico (Login)

CU01-e03	Acceso a recurso protegido con certificado electrónico		
Actor:	Usuario autorizado		
Descripción:	Procedimiento para el acceso a un recurso protegido de una aplicación utilizando como información de autenticación un certificado electrónico		
Precondiciones:	1. El usuario no está autenticado 2. Desde el navegador es posible acceder al certificado del usuario, bien porque lo ha instalado o está accesible desde alguna tarjeta inteligente. 3. El usuario ha autorizado la cesión de datos al proveedor de servicios.		
Poscondiciones:	El usuario está autenticado en el sistema o recibe un mensaje de error.		
Flujo normal:	1	El usuario intenta acceder a un recurso protegido de la aplicación	
	2	Se obtiene la lista de certificados accesibles desde el navegador	
	3	a	Si la lista contiene uno o más certificados. El usuario elige un certificado entre la lista de certificados válidos.
	4	Se comprueba el certificado elegido por el usuario.	
	4	a	La información es correcta, se autentica el usuario y se accede al recurso protegido solicitado.
Flujos alternativos:	4	b	No es posible autenticar al usuario con el certificado elegido , se devuelve al usuario un mensaje de error
	3	b	Si no hay ningún certificado disponible, se muestra un mensaje de error “No hay certificados disponibles.”

Tabla 3 - Descripción del escenario e03 del caso de uso CU01

CU01-e04 Salir de la aplicación A (Logout)

CU01-e04	Logout	
Actor:	Usuario autorizado	
Descripción:	Procedimiento para dejar de autenticado en el sistema	
Precondiciones:	1. El usuario está autenticado 2. El usuario se encuentra en un recurso protegido (página) de una de las aplicaciones.	
Poscondiciones:	El usuario no está autenticado en el sistema.	
Flujo normal:	1	El usuario pulsa el enlace “Salir de aplicación”
	2	El usuario no está autenticado. Se navega a la página de inicio de la aplicación

Tabla 4 - Descripción del escenario e04 del caso de uso CU01

Plan de pruebas

A continuación se muestran el conjunto de pruebas que forman parte del plan de pruebas de aceptación del sistema, con las que se va a verificar que el sistema implementado cumple con la funcionalidad solicitada.

Casos de prueba

Se muestran los casos de prueba que permiten validar que el funcionamiento del sistema se corresponde con su especificación. Estos casos son aplicables a todas las aplicaciones que se implementen para validar el sistema de autenticación única.

Caso de prueba	CP01	Acceso a la sección pública de la aplicación A	
Precondiciones		El usuario no está autenticado en el SSO	
Acciones		1	Se intenta acceder a una página de la sección pública
Poscondiciones		Se navega a la página pública de la aplicación Se muestra un mensaje “Sección pública de la aplicación A”	

Tabla 5 - Especificación del caso de prueba CP01

Caso de prueba	CP02	Acceso a la sección privada de la aplicación A	
Precondiciones		El usuario no está autenticado en el SSO	
Acciones		1	Se accede a la página de inicio de la aplicación A
		2	Se pulsa en el enlace “Acceso a la zona protegida”
		3	Se introducen usuario y contraseña válidos
Poscondiciones		El usuario está autenticado Se navega a la página de inicio de la sección privada de la aplicación. Se puede acceder a la información del usuario (atributos)	

Tabla 6 - Especificación del caso de prueba CP02

Caso de prueba	CP03	Autenticación en la aplicación A -error usuario
Precondiciones		El usuario no está definido en el servicio de directorio
Acciones	1	Se accede a la página de inicio de la aplicación A
	2	Se pulsa en el enlace “Acceso a la zona protegida”
	3	Se introduce un usuario y/o una contraseña no válidos
Poscondiciones		Se muestra un mensaje de error “No es posible identificar al usuario”

Tabla 7 - Especificación del caso de prueba CP03

Caso de prueba	CP04	Autenticación en la aplicación A -error de contraseña
Precondiciones		El usuario esta no está autenticado en el SSO
Acciones	1	Se accede a la página de inicio de la aplicación A
	2	Se pulsa en el enlace “Acceso a la zona protegida”
	3	Se introduce un usuario válido y una contraseña incorrecta
Poscondiciones		Se muestra un mensaje de error “La contraseña no es válida.”

Tabla 8 - Especificación del caso de prueba CP04

Caso de prueba	CP05	Autenticación en la aplicación A con certificado
Precondiciones		El usuario no está autenticado en el SSO
Acciones	1	Se accede a la página de inicio de una aplicación
	2	Se pulsa en el enlace “Acceso a la zona protegida”
	2	Se selecciona un certificado de la lista de certificados válidos
	3	Se accede a la aplicación con un usuario y contraseña validos
Poscondiciones		El usuario esta autenticado Se navega a la página de inicio de la sección privada de la aplicación. Se puede acceder a la información del usuario (atributos)

Tabla 9 - Especificación del caso de prueba CP05

Caso de prueba	CP05	Salir de una aplicación
Precondiciones		El usuario está autenticado en el SSO El usuario está en una página protegida de la aplicación
Acciones	1	Se pulsa en el enlace “salir de aplicación” * Puede ser necesario cerrar la ventana de la aplicación (seguir indicaciones)
Poscondiciones		El usuario no está autenticado. Se navega a la página de inicio

Tabla 10 - Especificación del caso de prueba CP05

Caso de prueba	CP06	Comprobación de la autorización de cesión de datos de usuario	
Precondiciones		El usuario está autenticado en el SSO El usuario no ha autorizado previamente la cesión de sus datos	
Acciones		1	Se accede a la página de inicio de la aplicación A
		2	Se pulsa en el enlace “Acceso a la zona protegida”
		3	Se introducen unas credenciales válidas (usuario/contraseña o certificado)
		4	Se pulsa el botón “Rechazar”
Poscondiciones		Se bloquea la cesión de información de usuario a la aplicación	

Tabla 11 - Especificación del caso de prueba CP06

Caso de prueba	CP07	Comprobación del servicio balanceado de proveedor de servicio	
Precondiciones		El usuario está autenticado en el SSO El usuario ha autorizado previamente la cesión de sus datos El usuario se encuentra en una página restringida de la aplicación	
Acciones		1	Se desconecta de la subred interna el nodo del proveedor de servicio que ha atendido la petición actual del usuario
		2	Se espera 1 min (tiempo de comprobación de la conexión del balanceador para que detecte que está fuera de servicio)
		3	Se intenta acceder a un enlace en la zona protegida de la aplicación
Poscondiciones		Se ha creado un contexto de autenticación en uno de los nodos del proveedor de servicios disponibles.	

Tabla 12 - Especificación del caso de prueba CP07

Conjunto de datos de prueba

Para realizar la validación del sistema de autenticación única se ha definido un caso de ejemplo que permita la validación del sistema SSO. Sea el sistema a implantar el de una empresa de viajes y actividades de ocio especializada en experiencias extremas llamada "atrapados".

Se han definido las siguientes aplicaciones:

- **experiencias**: encargada de la gestión de actividades: planificador de actividades o experiencias que permite realizar la gestión comercial. Se define la siguiente funcionalidad:
  - registro de clientes
  - gestión de experiencias
  - aprobación de ofertas especiales
- **proveedores**: encargada de la gestión de proveedores (administración), que permite realizar la gestión de los proveedores para las actividades contratadas. Se define la funcionalidad:

- gestión de alojamientos
- gestión de transportes
- aprobación de gastos extraordinarios.

La siguiente tabla muestra usuarios definidos del sistema:

Nombre	Usuario	Tipo de empleado	Función/Puesto
Joyce Byers	Joyceb	Admin	Gerente
Jim Hopper	Jimh	Admin	Administrativo
Jane Hopper	janeh	admin	Responsable de administración
Michael Wheeler	mikew	comercial	Comercial zona norte-oeste
William Byers	willb	comercial	Responsable comercial
Dustin Henderson	dustinh	comercial	Comercial zona norte
Lucas Sinclair	Lucass	comercial	Comercial zona este

Tabla 13 - Usuarios definidos en el sistema

Se ha definido dos tipos de empleados: admin y comercial.

Y los siguiente roles del sistema: Responsable comercial, Responsable de administración y Gerente.

Respecto a los permisos de acceso a las funcionalidades de las aplicaciones:

Aplicación	Funcionalidad	Tipo empleado/Rol
Experiencias	registro de clientes	Comercial
	gestión de actividades	Comercial
	aprobación de ofertas especiales	comercial + Responsable comercial
Proveedores	gestión de alojamientos	Admin
	gestión de transportes	Admin
	aprobación de gastos extraordinarios	admin+ Responsable administración

Tabla 14 - Relación de usuarios / roles del sistema

3.3 Arquitectura del sistema

En diagrama se pueden identificar, a alto nivel, los componentes básicos que intervienen en el sistema y las principales relaciones entre sí:

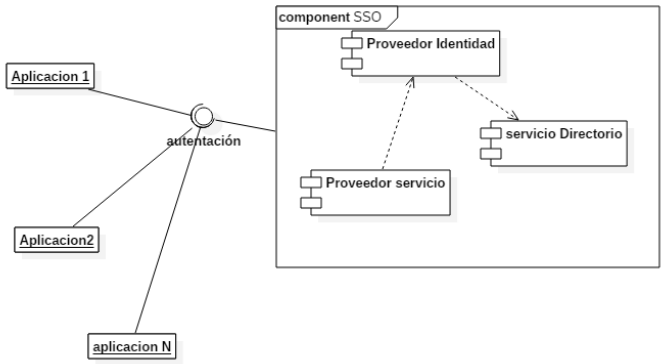


Figura 8 - Arquitectura del sistema

Respecto al proveedor de servicio, en la solución propuesta se ofrece a través de clúster de balanceo de carga. Este clúster está formado por proxy/balanceador y distintos nodos. De forma que este servicio puede escalar fácilmente según las necesidades. Las aplicaciones se despliegan en un contenedor de servlet en los nodos del proveedor de servicio.

El proveedor de identidad se conecta al servicio de directorio para verificar la identidad de los usuarios y consultar información de los mismos.

Respecto a la arquitectura de red, se sigue el diseño de red perimetral o DMZ, en la se utilizarán dos subredes, una pública y otra privada, separadas por dos firewall encargados de controlar el tráfico entre ellas. En la siguiente figura muestra el diagrama de red:

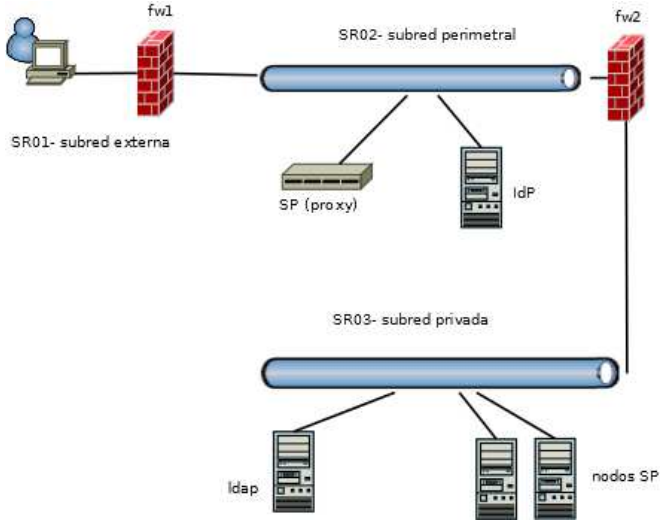


Figura 9 - Diagrama de red del sistema

En la subred perimetral se despliegan los servidores que han de ser públicos y accesibles por los usuarios de las aplicaciones. Por ejemplo, el proxy inverso/balanceador del servicio SP y el IdP. Mientras en la subred interna, se conectan los servidores que no son públicos. A estos servidores se accede a través de dispositivos en la subred perimetral. Por ejemplo, el servidor de ldap o los nodos del servicio balanceado de SP.

3.4 Diseño detallado e implementación

Componentes

Sistema de autenticación única (SSO)

Servicio de directorio

El servicio de directorio almacena la información de los usuarios de la organización. En la siguiente figura se muestra el modelo de datos del servicio:

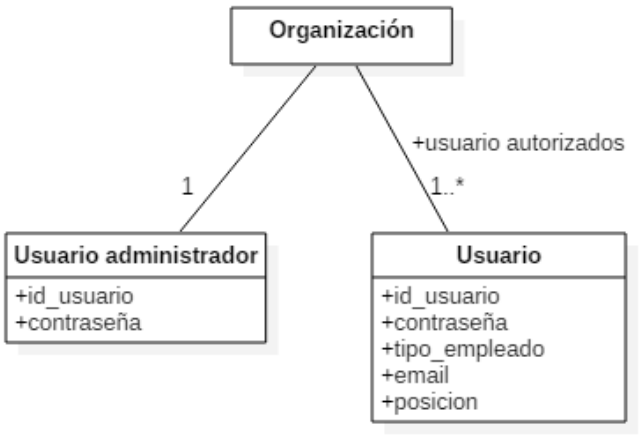


Figura 10 - Modelo lógico de datos del servicio de directorio

Como puede observarse, además de los usuarios que acceden a las aplicaciones, se crea un usuario con el que gestionar los datos (altas, bajas, modificaciones,…).

El servicio de directorio basa en OpenLDAP, una implementación libre y de código abierto del protocolo LDAPv3. Se ha utilizado la versión 2.4.44, la última versión disponible del repositorio oficial de centos 7.

Además de la configuración básica indicada para todos los equipos, respecto a este servicio:

- Se permite solo conexiones cifradas a través del puerto seguro 636 (ldaps). Se han generado los certificados electrónicos para el servidor ldap.

- Se han deshabilitado las conexiones anónimas (anonymous binds).

Carga de datos

Se muestran algunas de las entradas del servicio de directorio de la organización correspondientes a la definición del dominio de la organización, la unidad organizativa en la que se definen los usuarios y la entrada de uno de los usuarios definidos en dicho dominio:

```
dn: dc=atrapa,dc=dos
dc: atrapa
objectClass: top
objectClass: domain

dn: ou=profile, dc=atrapa, dc=dos
objectClass: organizationalUnit
ou: profile

dn: uid=willb, ou=profile, dc=atrapa, dc=dos
objectClass: inetOrgPerson
uid: willb
cn: William Byers
sn: Byers
userPassword: {SSHA}B80GI7hIH2L68w+8cwoVWjlezMpc6F9y
mail: willb @atrapa.dos
title: Responsable Comercial
employeeType: comercial
```

La entrada userPassword corresponde a la contraseña del usuario. Para almacenar la contraseña se ha utilizado la función resumen SSHA (variante de SHA con semilla), tal como indica el prefijo {SSHA} del valor del campo. Para generar el resumen correspondiente a una contraseña dada se utiliza el comando slappasswd.

Una vez definido el usuario de administración podemos gestionar los datos del directorio utilizando un cliente Ldap como, por ejemplo, LdapAdmin:

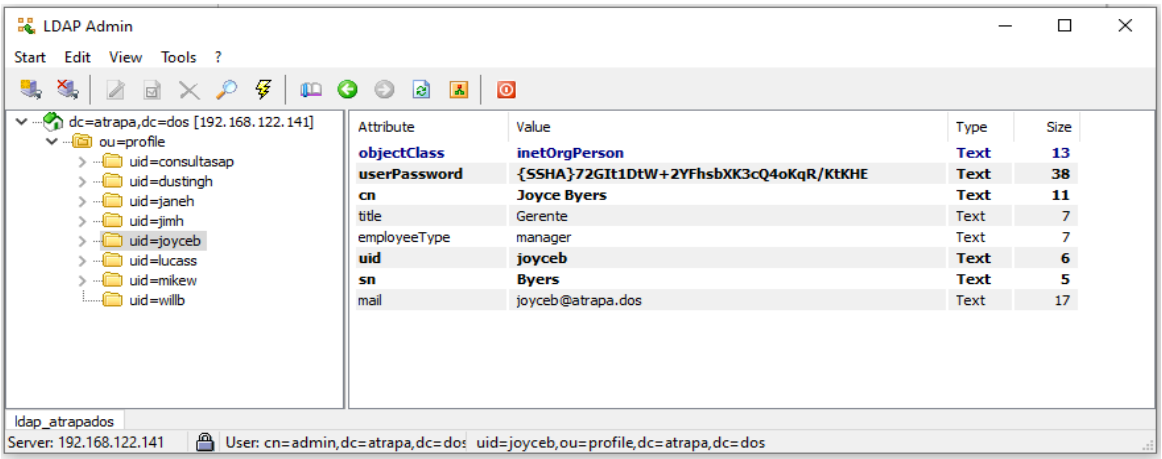


Figura 11 - Cliente Ldap (LDAP Admin)

Servicio IdP

La implementación del proveedor de identidad se basa en Shibboleth IdPv3.

Shibboleth IdP consiste principalmente en una aplicación web basada en la especificación 3.0. Funciona en la mayoría de los contenedores de servlet compatibles, si bien oficialmente sólo están soportados los contenedores Jetty (Jetty 9.2+) y Tomcat (Tomcat 8+) .

Este servicio está formado por los siguientes elementos:

- **servidor web apache**: actúa de frontend del componente recibiendo las peticiones de los usuarios y redirigiéndolas al contenedor de servlet a través de un conector AJP en el puerto 8009.
- **contenedor de servlet**: contenedor donde se despliega la aplicación web.
- **aplicación Shibboleth IdP (fichero war)**: aplicación web que es implementa las funciones de este servicio.

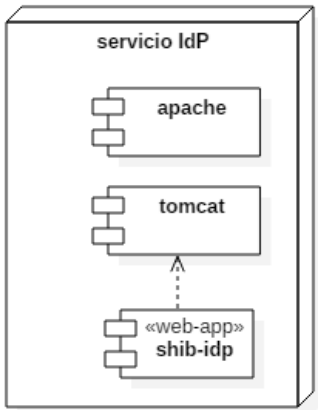


Figura 12 - Diagrama de componentes de Shibboleth IdP3

Apache

Respecto al servidor web, siguiendo los requerimientos de Shibboleth IdP v3 (anexo A) se ha instalado la versión 2.4.6 del servidor web apache. Se han definido dos host virtuales:

- El primero escucha en el puerto 80, realizando una redirección al puerto 443 para forzar el acceso a través de https.
- El segundo, escucha en el puerto 443, que procesa las peticiones de los usuarios.

Respecto a la seguridad, se ha habilitado SSL, indicando la localización de los certificados SSL. Se han seleccionado versiones seguras del protocolo (TLS1.2 o superiores) así como las suites de cifrado permitidas. Se muestran las principales directivas utilizadas en la configuración:

```
SSLEngine on
SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!SEED:!IDEA:!RC4:!LOW:!3DES:!kRSA
SSLHonorCipherOrder on
```

```
SSLCertificateFile /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
```

Como se ha indicado anteriormente, se define el reenvío de las peticiones del virtual host al contenedor de servlet Tomcat a través del puerto 8009 (utilizando el protocolo AJP).

Tomcat

Siguiendo los requerimiento de Shibboleth IdP v3 (Anexo A), se ha instalado la versión 8.5.46 de Tomcat y la versión de 1.8 de la distribución java de Oracle.

Se han configurado los conectores a Tomcat, en el fichero conf/server.xml:

- Se han comentado los conectores definidos por defecto ya que no se van a utilizar.
- Se ha definido un conector específico al que se reenviarán las peticiones recibidas por el servidor apache a través del puerto 8009:

```
<Connector port="8009" address="127.0.0.1" enableLookups="false"
redirectPort="443" protocol="AJP/1.3" tomcatAuthentication="false" />
```

Con el atributo address se permiten conexiones entrantes solo desde el propio equipo, para mayor seguridad.

Shibboleth IdP

Uno de los principales ficheros de configuración de Shibboleth IdP es el fichero *$IDP_HOME/conf/idp.properties.* En este se definen los siguientes parámetros del IdP:

idp.entityID : Identificador de IdP (entidad SAML)

```
idp.entityID=https://idp.atrapa.dos/idp/shibboleth
```

idp.authn.flows: Lista ordenada, por prioridad, de los mecanismos de autenticación que utilizará el IdP. Por defecto, solo se define la autenticación mediante usuario y contraseña.

```
idp.authn.flows=Password
```

idp.cookie.secure: El mecanismo de persistencia de sesiones, por defecto, está basado en cookies en el cliente. Con esta propiedad indicamos que las cookies deben cifrarse (sin cifrar por defecto).

```
idp.cookie.secure = true
```

idp.additionalProperties: Indica otros ficheros de configuración que ha de cargar:

```
idp.additionalProperties=/conf/ldap.properties, /conf/saml-nameid.properties,
/conf/services.properties, /conf/authn/duo.properties
```

Conexión a ldap

El proveedor de identidad se conecta al directorio de la organización para consultar los datos de los usuarios y realizar las verificaciones de identidad. Los ficheros de configuración involucrados son:

```
$IDP_HOME/conf/ldap.properties
$IDP_HOME/conf/authn/ldap-authn-config.xml
```

En el fichero *ldap.properties* se definen las propiedades de conexión a ldap:

```
idp.authn.LDAP.authenticator          = bindSearchAuthenticator
idp.authn.LDAP.ldapURL                = ldaps://ldap.atrapa.dos:636
idp.authn.LDAP.useStartTLS            = false
idp.authn.LDAP.useSSL                 = true

## Path del certificado del servidor ldap
idp.authn.LDAP.trustCertificates      = ${idp.home}/credentials/ldap.crt

## Atributos que se devolveran durante la autenticacion
idp.authn.LDAP.returnAttributes       = cn,title,employeeType,mail

## DN resolution properties ##
idp.authn.LDAP.baseDN                 = ou=profile, dc=atrapa, dc=dos
idp.authn.LDAP.userFilter             = (uid={user})
# configuracion de la consulta, usuario y clave del usuario de consulta
idp.authn.LDAP.bindDN         = uid=consultasap, ou=profile, dc=atrapa, dc=dos
idp.authn.LDAP.bindDNCredential       = <clave-usuario-consulta>
```

En /ldap-authn-config.xml se realiza la definición de los distintos Beans implicados en los distintos mecanismos de autenticación basados en Ldap. El sistema utiliza el autenticador *bindSearchAuthenticator*. Este autenticador se basa en la búsqueda de un usuario cuyo atributo *uid* sea el indicado en el campo usuario (*idp.authn.LDAP.userFilter*) y la contraseña sea la indicada en el atributo *userPassword* dentro de la rama *ou=profile, dc=atrapa, dc=dos* del directorio.

En el atributo idp.authn.LDAP.trustCertificates se indica la localización del certificado del servidor ldap necesario para establecer la conexión segura.

Registro del proveedor de servicios

Para poder utilizar los servicios de un proveedor de identidad, el proveedor de servicios ha de estar registrado en él. En este proceso el proveedor de identidad obtiene los metadatos del proveedor de servicio a través de un proveedor de metadatos. Esta definición se realiza en el fichero $IDP_HOME/conf/metadata-providers.xml. FilesystemMetadataProvider permite cargar los metadatos del proveedor de servicio desde un fichero local, previamente descargado desde SP:

```
<MetadataProvider id="LocalMetadata"  xsi:type="FilesystemMetadataProvider"
metadataFile="%{idp.home}/metadata/sp_atrapados-metadata.xml"/>
```

Servicio Proveedor de servicios

El proveedor de servicios recibe las peticiones de acceso a los recursos de las aplicaciones, las intercepta, y comprueba los requisitos de acceso. Si es necesario, hace la petición al proveedor de identidad para que verifique la identidad del usuario. Una vez satisfechas las necesidades de control de acceso la petición sigue su procesamiento.

El proveedor de servicio Shibboleth SP v3 para Linux está formado por tres elementos:

- servicio shibd.
- servidor web: se utiliza apache 2.4 versión siguiendo los requerimientos de Shibboleth SP.
- mod_shib: módulo de apache que implementa parte de la funcionalidad específica del proveedor de servicio y permite la interacción de los elementos anteriores.

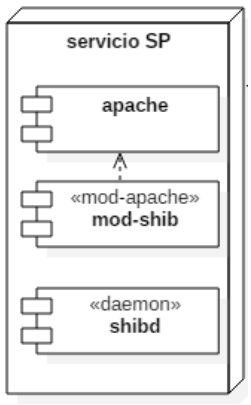


Figura 13 - Diagrama de componentes del proveedor de servicios

De forma adicional, en el mismo servidor se ha instalado un contenedor de servlet, Tomcat v.8.5.46, donde se han desplegado las aplicaciones de usuario.

servidor web

Respecto al servidor web, siguiendo los requerimientos de Shibboleth SP se utiliza apache en modo worker MPM, que implementa un servidor híbrido multiproceso-multihilo. Para ello se ha modificado el fichero */etc/httpd/conf.modules.d/00-mpm.conf*, para sustituir el módulo prefork MPM utilizado defecto:

```
# LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
LoadModule mpm_worker_module modules/mod_mpm_worker.so
```

Shibboleth SP

La configuración del proveedor de servicio se realiza a través de los ficheros:

- en el directorio */etc/Shibboleth*, siendo shibboleth2.xml uno de los más importantes.
- /etc/httpd/conf.d/shib.conf en el que se configura el módulo de apache mod_shib.

En *shibboleth2.xml* se definen los atributos más importantes de la configuración del SP.

Dentro de este fichero, se declara el elemento *ApplicationDefaults* en el que se indica*:*

- el atributo *entityID,* permite definir el identificador de identidad SAML (SAML entityID) utilizado por el proveedor de servicio en las comunicaciones SAML:

```
entityID="https://sp.atrapa.dos/shibboleth"
```

- El atributo cipherSuites, que define las suites de cifrado permitidas:

```
cipherSuites="DEFAULT:!EXP:!LOW:!aNULL:!eNULL:!DES:!IDEA:!SEED:!RC4:!3DES:!kRS
A:!SSLv2:!SSLv3:!TLSv1:!TLSv1.1"
```

Además del elemento *ApplicationDefaults* se definen otros elementos importantes:

- Session para la configuración de sesiones de autenticación, donde podemos indicar el uso de cookies cifradas (atributo cookieProps), tiempo de vida de la sesión del SP (timeout), tiempo de inactividad de la sesión, mecanismo de gestión de sesiones, etc:

```
  <Sessions lifetime="28800" timeout="3600"
      relayState="ss:mem"  checkAddress="false" handlerSSL="true"
      cookieProps="https" consistentAddress="true">
```

- SSO que permite la configuración de la funcionalidad single-sign on para el IdP utilizado:

```
<SSO entityID="https://idp.atrapa.dos/idp/shibboleth"
     discoveryProtocol="SAMLDS"
     discoveryURL="https://idp.atrapa.dos/DS/WAYF"> SAML2 SAML1 </SSO>
```

- MetadataProvider: para indicar la localización de los metadatos del IdP, y permitir su registro en el proveedor de servicio. Puede realizarse de forma estática a través de un fichero local (previamente descargado) o de forma dinámica a través de una url:

```
<MetadataProvider type="XML" validate="true" path="idp-atrapa.xml"/>
```

Aplicaciones

Para validar el sistema de autenticación se han implementado dos aplicaciones, siguiendo el conjunto de datos de prueba descrito en el plan de pruebas del sistema. Cada aplicación está formada por una sección pública y una privada.

La sección pública está formada por una página, la página de inicio de la aplicación, que puede accederse sin restricción de acceso.

La sección privada estará formada por un conjunto de páginas, una de inicio y otras que implementan las distintas funcionalidades de la aplicación.

La figura muestra el diagrama de la interfaz de usuario de las aplicaciones web:

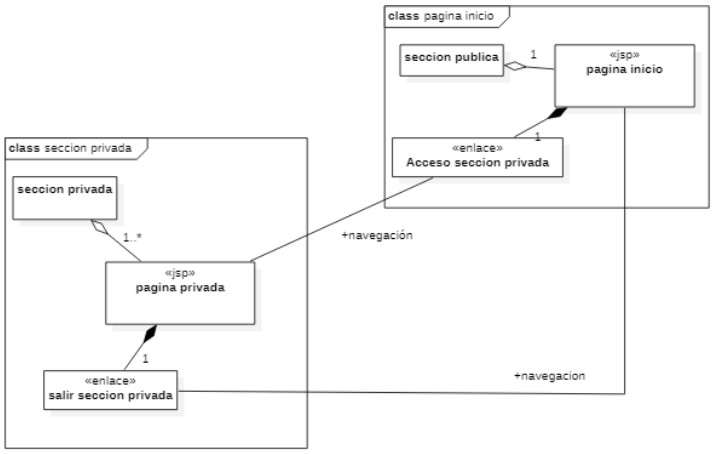


Figura 14 - Diagrama del interfaz de usuario de las aplicaciones web

Se han implementado dos aplicaciones web desarrolladas en java y han sido desplegadas en un contenedor de servlets Tomcat 8. Tomcat se ha instalado en el mismo servidor del proveedor de servicios encargado de controlar el acceso a las aplicaciones.

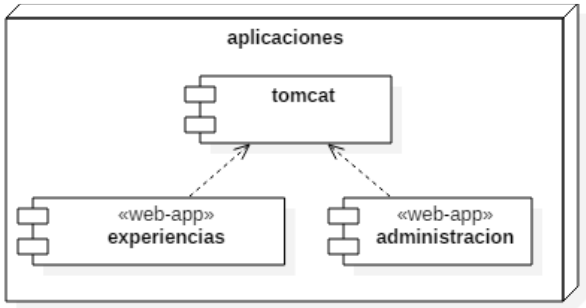


Figura 15 - Diagrama de componentes del servidor de aplicaciones

Las peticiones de acceso a los recursos de la aplicación, una vez procesadas por el proveedor de servicios, se reenvían a Tomcat través de un conector AJP en el puerto 8009.

Las URLs de acceso a las dos aplicaciones son:

- proveedores: https://sp.atrapa.dos/proveedores
- experiencias: https://sp.atrapa.dos/experiencias

En cada una de estas aplicaciones existe:

- sección pública: formada por una página index.jsp que es la página de inicio de la aplicación:

Figura 16 - Página de inicio (zona pública) de la aplicación proveedores

- una sección privada: formada por cuatro páginas entre las que se puede navegar. La URL de acceso a la sección privada es:
  - https://sp.atrapa.dos/proveedores/secure
  - https://sp.atrapa.dos/experiencias/secure

Las páginas de la zona protegida muestran información del usuario conectado obtenida desde el proveedor de identidad (email, posición en la empresa).

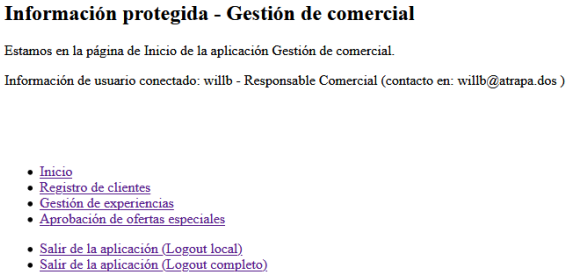


Figura 17 - Página de inicio de la aplicación experiencias

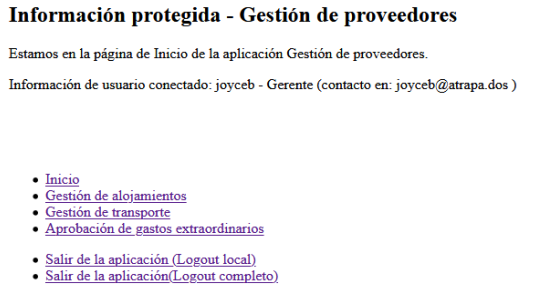


Figura 18 - Página de inicio de la aplicación proveedores

Registro de aplicaciones en el SP

A nivel de proveedor de servicios no existe el concepto de aplicación, solo entiende de recursos (URLs) cuyo acceso es necesario controlar. Las peticiones de ciertas URLs serán interceptadas por apache y procesadas por el módulo mod_shib, según la configuración.

Se muestra la configuración para controlar el acceso a la aplicación de Gestión de proveedores, a través de la url */proveedores/secure* en el fichero */etc/httpd/conf.d/shib.conf*:

```
<Location /proveedores/secure>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require shib-session
</Location>
```

AuthType indica que la petición la procesará el Shibboleth, y la directiva ShibRequestSetting indica que cualquier acceso debe autenticarse contra el IdP.

Una vez comprobada y realizada la autenticación, se continúa el procesamiento de la misma por apache. En este caso, como se ha indicado, se han desplegado en contenedor de servlet

Tomcat instalado en el mismo servidor, por lo que la petición se reenvía a través de un conector AJP en el puerto 8009.

Como se ha comentado, no es necesario realizar ninguna configuración en la aplicación. Apache intercepta las peticiones del usuario y el SP comprueba los requisitos de control de acceso. Es importante tener esto en cuenta respecto del uso de conexiones no cifradas. Por ejemplo, si al proporcionar un servicio balanceado se finaliza la conexión SSL en el balanceador. Si se despliegan las aplicaciones fuera de apache, desde la subred privada se podrían acceder a las aplicaciones sin que se apliquen los controles de acceso. En estos casos será necesario tomar medidas adicionales para evitar esta situación.

Autenticación mediante certificado electrónico

En el proceso de autenticación mediante certificado electrónico, la gestión de los mismos se realiza en el proveedor de identidad, encargado de realizar la autenticación de los usuarios.
La autenticación por certificado electrónico está implementada en Shibboleth IdP v3. Para poder utilizarla es necesario habilitarla, indicando el identificador del método de autenticación en el atributo idp.authn.flows, en el fichero *conf/idp.properties:*

```
idp.authn.flows=X509|Password
```

De esta forma se intenta autenticar al usuario mediante certificado electrónico y, si no es posible, mediante usuario y contraseña. Para asegurarse que la información del certificado que se transmite, a través de la conexión HTTPS desde el navegador, llega a la aplicación es necesario definir en el host virtual que procesa las peticiones del usuario las siguientes directivas:

```
<Location /idp/Authn/X509>
  SSLVerifyClient optional_no_ca
  SSLOptions -StdEnvVars +ExportCertData
</Location>
```

Con esta configuración, se indica que es requerido el uso de certificado electrónico por parte del usuario pero que no es necesario que se verifique la autoridad certificadora, ya que en la implementación actual los certificados de los usuarios son autofirmados.
Por otra parte, con la opción +ExportCertData se consigue que la información del certificado se reenvíe a través dela conexión AJP como variables de sesión SSL_*, de forma sean transmitidas hasta la aplicación Shibboleth IdP (Tomcat).

Para cada usuario se ha generado un certificado electrónico, con el atributo cn igual al uid del usuario.

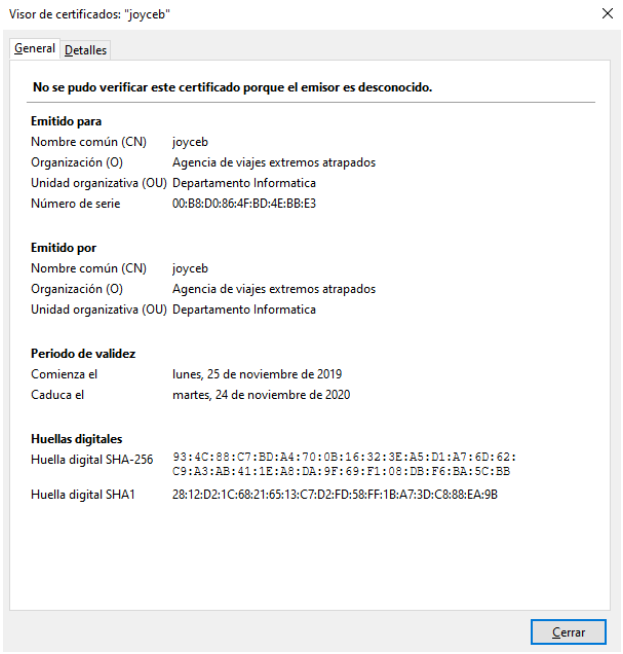


Figura 19 - Certificado de usuario generado para joyceb

Recuperación de la información del usuario

Una de las características más destacables de Shibboleth es la capacidad de extraer y exponer a las aplicaciones información del usuario desde el repositorio de la organización. Además, la información recuperada puede ser gestionada en función del proveedor de servicio.

Configuración IdP

Respecto al IdP, en el fichero $IDP_HOME/conf/*attribute-resolver.xml* se realiza la definición del conector de datos y la definición de los atributos a recuperar.

El conector utilizado es de tipo LDAPDirectory y permite extraer los datos del servicio de directorio dados los parámetros de conexión:

```
<DataConnector id="ldapConnector" xsi:type="LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:false}"
        trustFile="%{idp.attribute.resolver.LDAP.trustCertificates}" >
```

A continuación de se muestra la definición del atributo tipo de empleado:

```
<AttributeDefinition xsi:type="Simple" id="employeeType">
 <InputDataConnector ref="ldapConnector" attributeNames="employeeType"/>
 <AttributeEncoder xsi:type="SAML1String"
    name="urn:mace:dir:attribute-def:employeeType" encodeType="false" />
 <AttributeEncoder xsi:type="SAML2String"
    name="urn:oid:2.16.840.1.113730.3.1.4" friendlyName="employeeType"
    encodeType="false" />
</AttributeDefinition>
```

De forma similar han sido definidos el resto de atributos recuperados del directorio.

El elemento PolicyRequirementRule dentro de `AttributeFilterPolicy` en el fichero $IDP_HOME/conf/attribute-filter.xml permite definir a qué proveedores de servicio se va a exponer los datos. Con el atributo xsi:type se aplica a todos los proveedores de servicios (valor "ANY") o solo unos indicados (valor “OR”):

```
<AttributeFilterPolicy id="FPpublica">
  <PolicyRequirementRule xsi:type="ANY" />
    <AttributeRule attributeID="uid" permitAny="true" />
    <AttributeRule attributeID="mail" permitAny="true" />
…
```

Configuración en el SP

En la configuración del módulo mod_shib se incluyen directivas para controlar cómo la información enviada por el IdP se hace visible a la aplicación. La configuración para que la información sea expuesta a través de cabeceras HTTP:

```
ShibUseHeaders On
ShibUseEnvironment Off
```

Por último, la información que llega al proveedor de servicio a través de aserciones SAML ha de ser decodificada antes de ser utilizada. Para ello se asocia cada atributo con su tipo en el fichero *attributes-map.xml*:

```
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.1" id="uid"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
    <Attribute name="urn:oid:2.5.4.12" id="title"/>
    <Attribute name="urn:oid:2.16.840.1.113730.3.1.4" id="employeeType"/>
    <Attribute name="urn:mace:dir:attribute-def:cn" id="cn"/>
```

Consentimiento del usuario

Antes de realizar la cesión de la información a la aplicación, es necesario que sea autorizada la cesión de datos. Si no se aprueba la cesión se cancelará y no se expondrá información a las aplicaciones. En la configuración por defecto es el usuario quien realiza la autorización, siendo a opción más adecuada en general. Pueden establecerse otros mecanismos de autorización.

El comportamiento respecto al consentimiento para la cesión de datos se define a través del atributo p:postAuthenticationFlows del bean shibboleth.DefaultRelyingParty. Este bean se define en el fichero relaying_party.xml:

```
<bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
 <property name="profileConfigurations">
 <list>
   <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
..
```

Logout

En Shibboleth podemos encontrar dos tipos de logout:

- logout en el proveedor de servicio: en el que se elimina la información de sesión en el proveedor de servicio, a través de la url: *https://sp.atrapa.dos/Shibboleth.sso/Logout*

Cabe la posibilidad de utilizar el parámetro return para indicar la URL a la que invocar tras cerrar la sesión en el proveedor de servicios.

- logout en el proveedor de identidad: en el que se elimina la información de autenticación en el proveedor de identidad, a través del url: *https://idp.atrapa.dos/idp/profile/Logout*

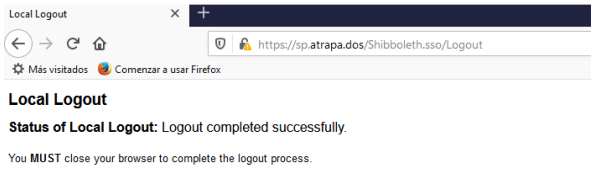


Figura 20 - Logout del proveedor de servicios

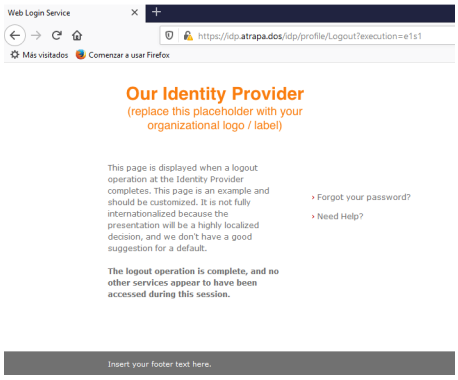


Figura 21 - Logout del proveedor de identidad

Es posible conseguir en una única acción conseguir realizar ambas opciones, utilizando el parámetro return, con la siguiente URL:

```
https://sp.atrapa.dos/Shibboleth.sso/Logout?return=https://idp.atrapa.dos/idp/profile/Logout
```

Tras invocarse el logout en el proveedor de servicios, se invoca en el proveedor de identidad y, si no hay errores en el procesamiento de estas peticiones, se consigue el logout en todos los sistemas.

Alta disponibilidad

Tal como se ha indicado, en la primera iteración se ha obtenido un sistema que funcionalmente cumple los requerimientos principales. Siguiendo la planificación inicial, en la segunda iteración se han añadido características al sistema que han permitido mejorar su disponibilidad y escalabilidad.

Respecto al proveedor de servicios, el sistema proporciona un servicio balanceado, a través de un clúster de balanceo de carga, formado un balanceador/proxy (HAProxy) y un backend formado por dos servidores, cuya configuración se detalla en el siguiente apartado.

Respecto al proveedor de identidad, por restricción de memoria del host, no ha sido posible poner en marcha un servicio balanceado de similares características al del proveedor de servicio, si bien su implementación sería muy similar.

Por último indicar, que si bien el sistema propuesto mejora las características de disponibilidad del sistema, está lejos de alcanzar una configuración en alta disponibilidad. En las configuraciones en alta disponibilidad se han de eliminar los puntos únicos de fallo del sistema. Si bien, por el propio diseño de algunos componentes de los productos de Shibboleth no es posible alcanzar una configuración cien por cien en alta disponibilidad (ni failover), es posible realizar mejoras al sistema implementado. En el apartado de líneas futuras de trabajo se indican algunas propuestas al respecto.

Servicio Proxy

Para la implementación del proveedor de servicio se ha desplegado un clúster de balanceo de carga, formado por un backend de dos nodos. Esta configuración permite ajustar a las necesidades del sistema añadiendo mas o menos nodos al backend.

Se ha utilizado como proxy inverso la versión 1.9.7 de HAProxy. Se ha configurado en modo SSL-pass-through, de forma que las conexiones https entrantes por el puerto 443 son procesadas y reenviadas a los servidores backend definidos, sin finalizar la conexión SSL. Este modo de operación, si bien limita la funcionalidad de procesamiento de las peticiones por parte de HAProxy, permite mantener la conexión cifrada hasta el servidor que la procesa en la subred interna, por lo que es más segura.
La configuración se realiza en el fichero /etc/haproxy/ haproxy-1.9.7.conf.

Se ha definido el frontend para recibir las peticiones de los usuarios, en los puertos indicados, y reenviar al backend bk_sp:

```
frontend ft_https
  bind *:80
  bind *:443
  mode tcp
  log global
  default_backend bk_sp
```

Respecto al backend definido, está formado por dos nodos o servidores, cuyo estado es monitorizado por HAProxy. El algoritmo de asignación de peticiones a los distintos nodos es el de menor número de peticiones, para tratar de repartir la carga entre los distintos nodos. Como se ha indicado, el proveedor de servicio de Shibboleth es una aplicación con estado. Por cuestiones de diseño no es posible compartir el estado entre los distintos nodos, por lo que se mantiene cierta afinidad entre los usuarios, concretamente su ip, y el servidor en el backend que trata la petición. Esta asociación se mantiene durante diez minutos tras los que se vuelve a aplicar el algoritmo de reparto de peticiones.

Se muestra la configuración del backend definido:

```
backend bk_sp
 # modo tcp ya que la conexion esta cifrada
 mode tcp
 log global
 balance leastconn
 stick-table type ip size 1m expire 10m
 stick on src

 server s1  sp1.atrapa.dos:443 check
 server s2  sp2.atrapa.dos:443 check
```

Infraestructura

Finalmente, se muestran los detalles de la infraestructura sobre la que se ha implementado la solución. Se ha utilizado como software de virtualización Oracle VirtualBox versión 6.0.14.

El equipo huésped que va a soportar la infraestructura tiene las siguientes características:

- Intel core i5-4670K 3.40GHz
- 8 vCPUs
- Arquitectura de 64bits
- 8GB RAM
- 1TB

Servidores

Se han creado las siguientes máquinas virtuales:

Nombre	Memoria	Disco	CPUs	S.O.	Servicios
idp.atrapa.dos	1,5 GB	20GB	2 vCPUs	CentOSv7	Proveedor de identidad (IdP)
ldap.atrapa.dos	512MB	10GB	1 vCPU	CentOSv7	Servicio de directorio
sp1.atrapa.dos	1 GB	20GB	2 vCPUs	CentOSv7	Proveedor de servicios 1(SP1)
sp2.atrapa.dos	1 GB	20GB	2 vCPUs	CentOSv7	Proveedor de servicios 2 (SP2)
sp.atrapa.dos	512MB	10GB	1 vCPU	CentOSv7	Proxy inverso/Balanceador
fw1.atrapa.dos	512MB	10GB	1 vCPU	CentOSv7	Cortafuegos 1

fw2.atrapa.dos	512MB	10GB	1 vCPU	CentOSv7	Cortafuegos 2

Tabla 15 - Lista de máquinas virtuales desplegadas

El dimensionamiento de memoria de los servidores se ha visto condicionado por las limitaciones de memoria del equipo huésped.

En todos los equipos se ha realizado una configuración inicial básica que consistido principalmente en la instalación de Centos v7 y unas medidas básicas de securización:

- se han eliminado los servicios no necesarios, por ejemplo postfix en el puerto 25.
- Se ha deshabilitado IPv6 ya que no es necesario.
- Se han configurado las distintas interfaces de red de cada dispositivo.
- Se ha actualizado el software instalado.
- Se ha securizado el servicio ssh: se ha limitado las conexiones del usuario root, se ha limitado el tiempo de inactividad de la sesión a 5 min,...

Adicionalmente en cada servidor se han configurado y securizado los distintos servicios que ofrece.

Infraestructura de red

En la siguiente figura se muestra el esquema de red de la solución implementada:

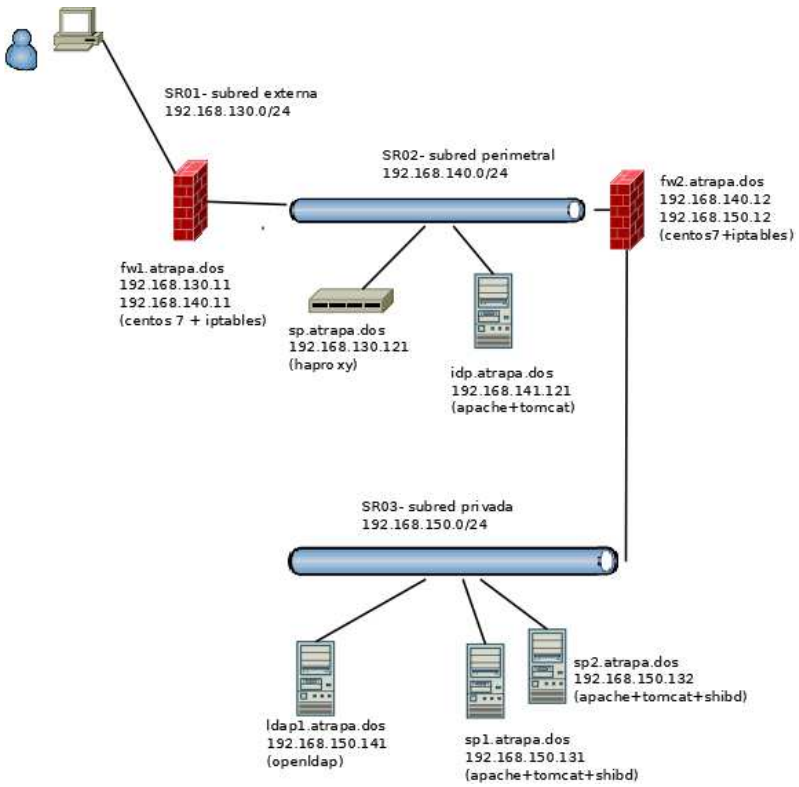


Figura 22 - Diagrama de red del sistema

Más detalles de la infraestructura de red desplegada pueden encontrarse en el anexo B de la memoria.

3.4 Validación

Se muestra el resultado obtenido de los distintos casos de prueba definidos en la fase de análisis que permiten validar que el sistema se ajusta a los requisitos establecidos.

Estos casos de prueba se han realizado en las dos aplicaciones implementadas utilizando versiones actualizadas de los navegadores Mozilla Firefox (v 71.0 -64-bit-), Microsoft Explorer (v 11.0.9600.19572) y Google Chrome (v 79.0.3945.88 -64 bit-). Es necesario que estén habilitadas las cookies propias (no las de terceros).

CP01	Acceso a la sección publica de la aplicación proveedores

Resultado: Correcto. Se accede a la sección pública sin necesitar estar autenticado.

Figura 23 - Página del área pública de la aplicación de Gestión de proveedores

CP02	Acceso a la sección privada de la aplicación proveedores

Resultado: Correcto. Tras la autenticación se accede a la página de inicio de la aplicación. Y es posible navegar por las páginas de la aplicación o de otras aplicaciones sin autenticarse.

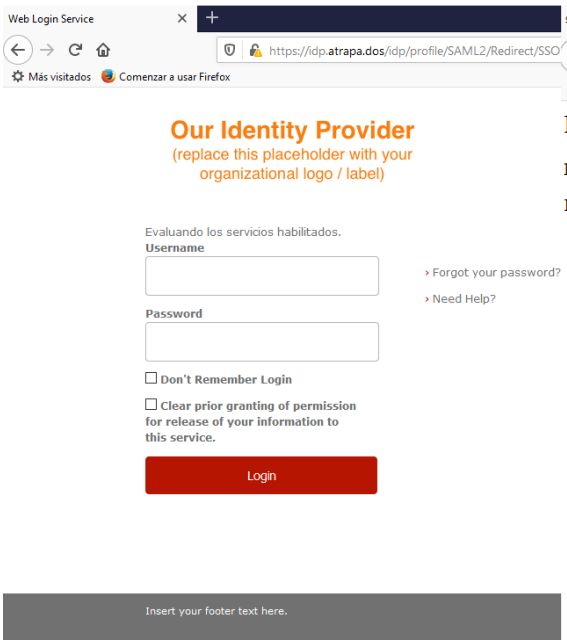


Figura 24 - Pagina de login del IdP

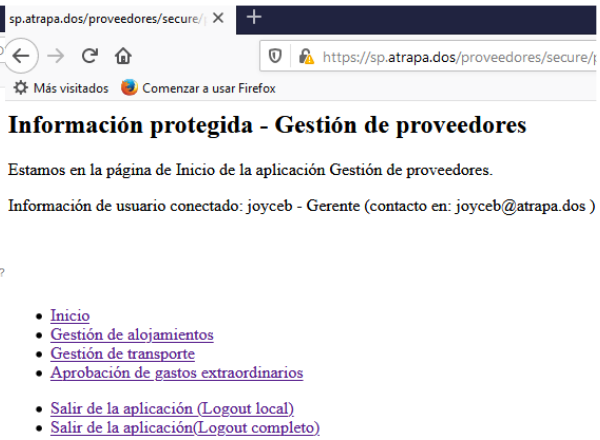


Figura 25 - Página de inicio de la aplicación proveedores

CP03/CP04	Autenticación en la aplicación proveedores -error usuario o contraseña

Resultado: Correcto. Tras introducir una combinación de nombre de usuario y/o contraseña erróneos, se devuelve un mensaje da error.

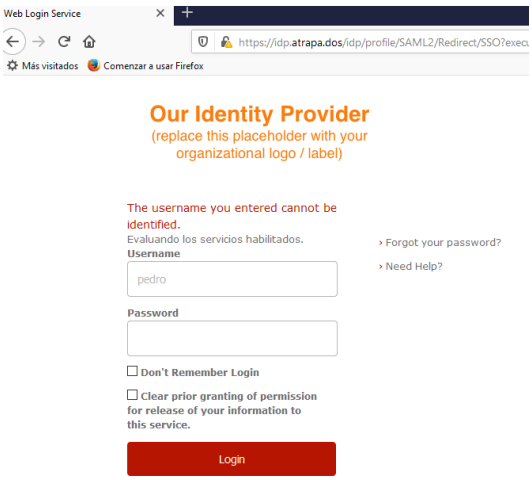


Figura 26 - Mensaje de error en página de login tras error de autenticación

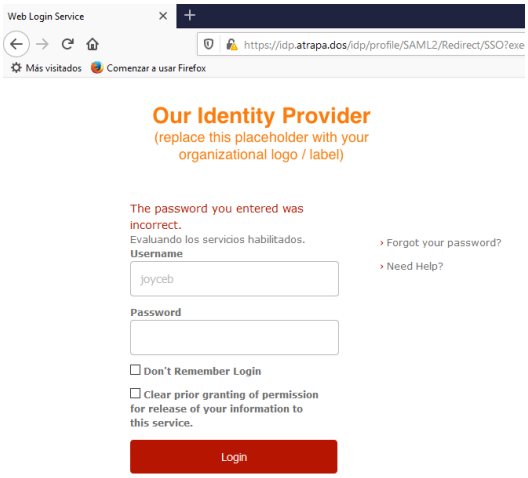


Figura 27 - Mensaje de error en página de login tras error en la contraseña

CP04	Autenticación en una aplicación con certificado

Resultado: Correcto. Se autentica el usuario con su certificado electrónico y accede a la aplicación.

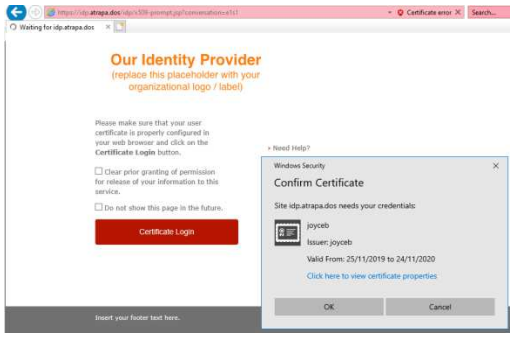


Figura 28 - Página de login con certificado electrónico y selección del mismo

CP05	Salir de una aplicación (logout local)

Resultado: Correcto. Se produce el logout de la aplicación (es necesario cerrar la ventana). Se invalida la sesión en el proveedor de servicios pero no la del proveedor de identidad.

Figura 29 - Página de logout de la aplicación

CP06	Autenticación de usuario sin autorización de cesión de datos

Resultado: Correcto. Si no se autoriza la cesión de información del proveedor de identidad al de servicios se cancela la cesión de datos:

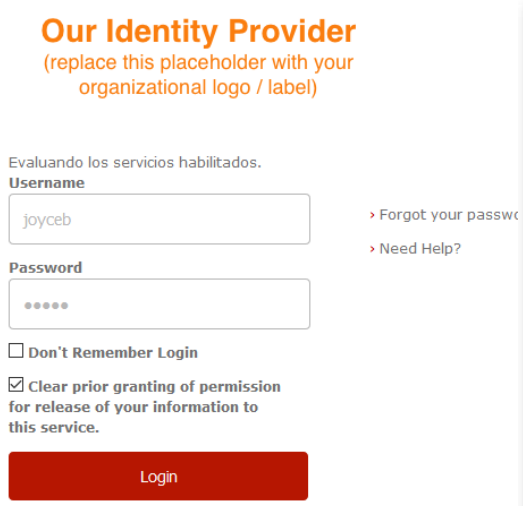


Figura 30 - Pagina de login (forzando pedir autorización)

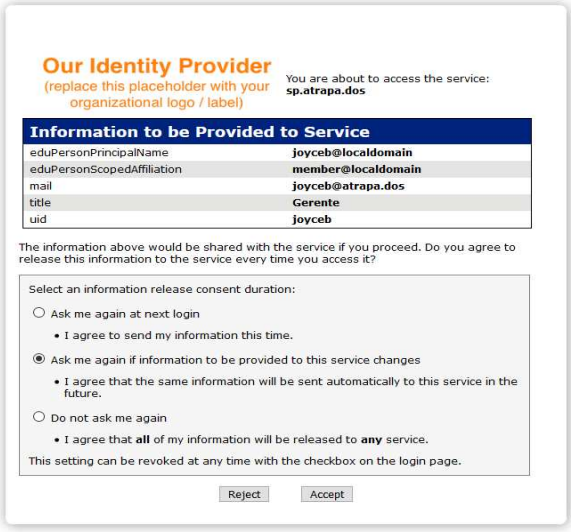


Figura 31 - Pagina de concesión de autorización

CP07	Comprobación del servicio balanceado del componente proveedor de servicio

Resultado: Correcto. Tras desconectar el nodo que ha procesado la petición del usuario.

Debido a la asociación de afinidad (sticky sesión), las peticiones siguientes se reenvían al mismo nodo. Una vez transcurrido un minuto, que es el tiempo de comprobación del enlace establecido en el balanceador, éste deshabilita al nodo fuera de servicio y redirige la petición a otro nodo del backend. La petición tarde algo más en procesarse ya que se ha crear el contexto de seguridad en el proveedor de servicio tras consultar al proveedor de identidad.

4. Conclusiones

4.1 Conclusiones del Trabajo

Una vez finalizada la validación del sistema implementado es momento de extraer algunas conclusiones del trabajo realizado.

Respecto a la metodología utilizada, el desarrollo iterativo e incremental se ha adaptado bastante bien al sistema de entregas parciales. Y ha permitido tener un producto funcional con un plazo suficiente para abordar nuevas iteraciones, que mejoren el producto, dentro de los plazos establecidos. La realimentación obtenida tras la primera iteración ha sido muy beneficiosa para abordar el resto del proyecto.

Los productos de Shibboleth proporcionan una base adecuada para el desarrollo de este tipo de sistemas. Son bastante configurables, adaptando su comportamiento y funcionalidad a las necesidades del sistema. Shibboleth desarrolla gran parte de su funcionalidad utilizando Spring Web Flow. Esta es una especificación de Spring que permite definir flujos declarativos de alto nivel. Esta permite de forma sencilla la definición funcionalidad adicional, que se integra adecuadamente en el sistema. Por ejemplo, definir nuevos flujos de autenticación o combinación de los existentes. Además esta funcionalidad es utilizable en otros escenarios, por ejemplo en otras aplicaciones webs.

Respecto a gestión de atributos, uno de los elementos más destacables de Shibboleth, comentar que se permite de forma sencilla exponer a las distintas aplicaciones integradas en el sistema la información de usuario almacenada en el directorio, como otras fuentes. Por otra parte, mencionar el control que posee el usuario sobre la transmisión de su información a las distintas aplicaciones y proveedores de aplicaciones.

Respecto al sistema implementado, si bien cumple con los requisitos definidos inicialmente, restricciones en la infraestructura sobre la que se ha implementado no han permitido desarrollar algunas características adicionales. Estos aspectos así como funcionalidad adicional son propuestos en el siguiente apartado en el que se ofrecen distintas propuestas para continuar el trabajo realizado que permitirían mejorar la solución propuesta.

4.2 Mejoras / Futuras aportaciones

En este apartado se abordaran algunas líneas en las que puede continuarse el trabajo que se ha realizado. En primer lugar se van a comentar algunas propuestas para mejorar la disponibilidad del sistema. A continuación se mencionan algunas mejoras que podrían realizarse para extender la funcionalidad de autenticación. Y por último otros aspectos en los podría mejorarse de cara a su puesta en explotación real.

Mejoras respecto de la alta disponibilidad

La solución implementada pretende mostrar las capacidades de un mecanismo de autenticación única, en este caso Shibboleth, y proporcionar una configuración adecuada. Como hemos visto en los sistemas en los que se implementan mecanismos de autenticación única, los elementos que lo forman se convierten en un elemento crítico de la infraestructura.

En esta línea se proponen algunas mejoras para cada uno de los componentes del sistema, que en definitiva, tratan de eliminar los distintos puntos de fallo del sistema o aumentar su escalabilidad, para alcanzar un mayor grado de disponibilidad del sistema.

No hay que perder de vista, que es necesario llegar a una solución de compromiso entre las necesidades de disponibilidad y el coste del sistema. Hay que ser consciente de la imposibilidad de llegar al cien por cien de seguridad. Es imprescindible que en el análisis previo, antes de implantar nuevas medidas, nos aseguremos que el nivel de disponibilidad del sistema final es mayor que el anterior. Es fácil obviar que al añadir nuevos componentes al sistema estamos aumentando su complejidad, y probablemente aumentan el número de puntos de fallo únicos.

Respecto al IdP

Respecto al IdP, es interesante la creación de un clúster de balanceo de carga, en el que las distintas peticiones se reparten entre múltiples instancias del proveedor de identidad. En esta configuración es necesario mantener cierta afinidad entre los usuarios y el servidor (stickness), ya que no es posible compartir estado entre los nodos. Es posible a través de sticky sessions en el balanceador, de forma similar a la implementación del proveedor de servicios.

Respecto al SP

Una alternativa que podría permitir mejorar la escalabilidad del proveedor de servicios, es la de desplegar las aplicaciones en un servidor distinto al del proveedor de servicios. El proveedor

de servicio recibe las peticiones, realiza la gestión del acceso y redirige la petición mediante conector AJP a un contenedor Tomcat desplegado en otro servidor dentro de la subred privada. En este caso, sería adecuado que la comunicación a través del conector AJP fuese cifrada para evitar, que desde la propia subred interna se pueda acceder a las aplicaciones sin pasar por el control de acceso. También habría que analizar la posibilidad de compartir el estado de las aplicaciones utilizando memcached o mecanismos similares.

Otra propuesta de mejorar consiste en la definición de un servidor de backup del servicio balanceado. Este básicamente permite mostrar mensajes de no disponibilidad del sistema en caso de caída, mantenimiento o incluso proveer un servicio de nivel degradado.

Respecto al servicio de directorio

Respecto al servicio de directorio, es posible utilizar un clúster de balanceo de carga para proporcionar la alta disponibilidad del servicio. Respecto a coherencia de la información, OpenLdap soporta distintas topologías que permiten la replicación de la información entre distintos servidores ldap, basadas principalmente en el uso del protocolo de sincronización de contenido de LDAP (LDAP Sync).

Respecto al balanceador

En una configuración de alta disponibilidad es necesario eliminar todos los puntos de fallo únicos. El balanceador se convierte en un punto de fallo único, por lo que sería conveniente tener respaldado el servicio de balanceo. El servicio se ofrece a través de una ip virtual, que es gestionada mediante keepalived o similar, en el que la alta disponibilidad se alcanza gracias a protocolos de enrutamiento como VRRP.

Mejoras respecto a la autorización

Como se ha indicado, uno de las desventajas de utilizar un sistema de autenticación única es que no proporciona una solución completa al problema del control de acceso a los recursos. De forma que en las aplicaciones han de establecerse mecanismos de control para comprobar si el usuario posee los permisos necesarios para acceder al recurso o aplicación (mecanismos de autorización). En la práctica, esto no supone un gran inconveniente, debido a que en la mayoría de las ocasiones la información de autorización (roles, permisos,…) es bastante dependiente de la propia aplicación y suele almacenarse junto a los datos de la aplicación, normalmente en una base de datos.

Sin embargo, en ocasiones, puede ser interesante poder gestionar desde el sistema SSO un primer nivel de autorización. Sea como ejemplo, en una institución universitaria en la que podemos distinguir varios colectivos como alumnos, docentes o personal de administración. Normalmente a cada colectivo se le ofrece un conjunto de servicios (aplicaciones) diferenciados. Por lo que podría ser interesante discriminar el acceso según el tipo de personal. Esta información podría almacenarse en el servicio de directorio, no es un dato que cambie frecuentemente, y podría ser utilizada para implementar para todas las aplicaciones un primer nivel de autorización. En cada aplicación se implementará un segundo nivel de control de acceso, más detallado y ajustado a sus necesidades.

Shibboleth SP permite utilizar información del usuario, ciertos atributos, para implementar este mecanismo de autorización de primer nivel, mediante el uso de directivas en la configuración del recurso en el módulo mod_shib de apache del proveedor de servicios.

Mejoras en otros aspectos

Otro aspecto interesante en el que se podría mejorar el trabajo realizado, de cara a la puesta en producción del sistema de autenticación, es la personalización de la interfaz de usuario de los distintos elementos, principalmente del proveedor de identidad:

- Páginas del flujo de navegación de los mecanismos de autenticación utilizados, como las páginas de solicitud de credenciales (usuario/contraseña, certificado electrónico,...).
- Página de inicio del proveedor de identidad
- Página de consentimiento de cesión de información a la aplicación/proveedor de servicio.
- Páginas de Logout.

Bibliografía

[1] Iterative and incremental development
https://en.wikipedia.org/wiki/Iterative_and_incremental_development

[2] Managing the development of large software system
http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf

[3] Autenticación de usuarios
http://www.rediris.es/cert/doc/unixsec/node14.html
Distributed and cloud computing. From parallel processing to the Internet of Things.

[4] Identity management
https://searchsecurity.techtarget.com/definition/identity-management-ID-management

[5] SAML 2.0
https://en.wikipedia.org/wiki/SAML_2.0

[6] Security Assertion Markup Language (SAML) V2.0 Technical Overview

[7]Shibboleth Architecture. Protocols and Profiles

[8] Conceptos Shibboleth
https://wiki.shibboleth.net/confluence/display/CONCEPT

[9] Shibboleth IDP v3
https://wiki.shibboleth.net/confluence/display/IDP30

[10] Shibboleth SP v3
https://wiki.shibboleth.net/confluence/display/SP3

[11] 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology
https://ieeexplore.ieee.org/document/159342

[12] OpenLDAP Directory services
http://www.openldap.org/doc/admin24/guide.html

[13] Autenticación de usuarios
http://www.rediris.es/cert/doc/unixsec/node14.html

[14] Escalabilidad
https://es.wikipedia.org/wiki/Escalabilidad

[15] Starter Guide HAProxy 1.9
http://cbonte.github.io/haproxy-dconv/1.9/intro.html

[16] Configuration Manual HAProxy 1.9
http://cbonte.github.io/haproxy-dconv/1.9/configuration.html

Anexos

Anexo A. Requerimientos de Shibboleth

Se indican los principales requerimientos de los productos de Shibboleth utilizados en la implementación del sistema de autenticación única.

Requerimientos Shibboleth IdP v3

Respecto a las distribuciones de Java están completamente soportadas las siguientes:

- Amazon Corretto 8 para Linux / Windows
- Amazon Corretto 11 para Linux/ Windows
- Oracle Java 8 para Linux / Windows (de Oracle Technology Network)
- Red Hat's OpenJDK 8 / 11 para Linux en Red Hat Enterprise Linux 7

La extensión criptográfica para Java (JCE) es necesaria, si bien, viene instalada por defecto para las distribuciones indicadas.
Se requiere un contenedor de servlet que implemente la versión 3.0.

Respecto al agente de usuario (navegador) no se tienen requerimientos específicos si bien se indica que las pruebas se han realizado sobre versiones recientes de los navegadores en los que se presuponen características HTML como almacenamiento local.
El IdP requiere del uso de cookies. No requiere el uso de cookies de terceros y no permite la incrustación del interfaz de usuario en frames de terceras partes.

Requerimientos Shibboleth SP v3

Shibboleth SP está escrito en C++ portable con algunas dependencias de librerías en C y C++.En teoría, es posible compilarlo en la mayoría de los sistemas operativos. Sin embargo solo se soporta oficialmente en las siguientes versiones de Windows, Linux y MacOS.
Respecto a Windows:

- Están soportadas las versiones Windows Server 2008 y posteriores. No esta soportados los sistemas basados en Itanium ni ARM.
- Se proporcionan instaladores para las versiones de 32 y 64bits. Si bien es posible compilarlo desde el código fuente, no se recomienda.

Respecto a Linux, estas soportadas oficialmente las siguientes distribuciones:

- Red Hat Enterprise and CentOS 6, 7, 8
- SUSE Linux Enterprise Server 12SP4

Respecto a Mac OS, solo ha podido realizar pruebas en las últimas releases 10.X y por tanto son las únicas soportadas oficialmente.

Shibboleth SPv3 requiere un servidor web, están soportados oficialmente:

- Apache 2.2 and 2.4 and FastCGI.
- Se recomienda el uso de apache worker MPM que implementa un servidor hibrido multihilo multiproceso.

Anexo B. Infraestructura de red desplegada

Para desplegar las tres subredes se han definido en virtual box, un interfaz de red anfitrión y dos redes internas con las siguientes características:

Subred	Nombre VB	Tipo VB	Dirección subred	Función
SR01	e0	Adaptador solo anfitrión	192.168.130.0/24	Subred externa
SR02	i0	Red interna	192.168.140.0/24	Subred perimetral
SR03	i1	Red interna	192.168.150.0/24	Subred interna

Tabla 16 - Lista de subredes del sistema

Para facilitar la configuración de los distintos equipos se han definido las siguientes subredes adicionales durante la etapa de instalación y configuración:

Subred	Tipo	Dirección de subred	Función
SRC01	Adaptador solo anfitrión	192.168.122.0/24	Subred de gestión
SRC02	NAT	10.0.3.0/24	Subred de acceso a internet

Tabla 17 - Lista de subredes de gestión

Respecto a los detalles del direccionamiento de la infraestructura utilizada, se proporcionan más detalles en el anexo B de la memoria.

Se han definido las direcciones IP de los servidores de la infraestructura en los ficheros de hosts: (/etc/hosts):

```
192.168.122.11  fw1g fw1g.atrapa.dos
192.168.122.111 px1g px1g.atrapa.dos
192.168.122.121 idpg idpg.atrapa.dos
192.168.122.131 sp1g sp1g.atrapa.dos
192.168.122.141 ldap1g ldap1g.atrapa.dos


192.168.140.11  fw1 fw1.atrapa.dos
192.168.140.111 sp.atrapa.dos
192.168.140.121 idp.atrapa.dos
192.168.140.12  fw2 fw2.atrapa.dos
```

solo en equipos con interfaces en la subred privada

```
192.168.150.12  fw2p fw2p.atrapa.dos
```

```
192.168.150.141 ldap ldap.atrapa.dos
192.168.150.131 sp1 sp1.atrapa.dos
192.168.150.132 sp2 sp2.atrapa.dos
```

De forma adicional se han añadido en los dispositivos de cada subred las reglas de encaminamiento necesarias para permitir la conectividad entre los equipos de las distintas subredes. Así en los dispositivos de SR01:

```
ip route add 192.168.140.0/24 via 192.168.130.11
```

No se define la ruta a la subred privada ya que desde la subred externa no se conoce la estructura interna.

En los dispositivos de SR02:

```
ip route add 192.168.130.0/24 via 192.168.140.11
ip route add 192.168.150.0/24 via 192.168.140.12
```

En los dispositivos de SR03:

```
ip route add 192.168.130.0/24 via 192.168.150.12
ip route add 192.168.140.0/24 via 192.168.150.12
```

Equipos comunicaciones

Se han definido dos equipos que actúan como enrutadores y cortafuegos, permitiendo las comunicaciones entre las distintas subredes.

Para su configuración, además de la configuración/securización básica:

- Se ha habilitado el reenvío de tramas.
- Las políticas por defecto de las cadenas INPUT, FORWARD y OUTPUT es DROP.
- Se han configurado las reglas según las necesidades y el uso de cada uno de los dispositivos.

FW1

Es el encargado de controlar el acceso de usuarios desde el exterior.

Permite el acceso a los servicios públicos ofertados por el sistema. En nuestro caso solo permite el acceso a través del puerto 80 y 443 a través del proxy.

FW2

Encargado de controlar el acceso a desde los servidores de la subred perimetral a la subred interna. Se definen reglas más específicas para permitir la comunicación entre:

- el proveedor de identidad y el servidor de ldap a través del puerto 636.
- el proxy y los backend del proveedor de servicio a través del puerto 443.